```
                        title   '8086 Bdos Interface, Jan, 1982'
                    ;*****************************************************************
                    ;*****************************************************************
                    ;**                                                             **
                    ;**  B a s i c   D i s K  O p e r a t i n g  S y s t e m        **
                    ;**          I n t e r f a c e   M o d u l e                    **
                    ;**                                                             **
                    ;*****************************************************************
                    ;*****************************************************************
                    ;
                    ;       Copyright (c) 1978, 1979, 1980,1981
                    ;       Digital Research
                    ;       Box 579, Pacific Grove
                    ;       California
                    ;
                    ;
                    ;       Jan 12, 1982
                    ;
                    ;
                    ;*********************************************
                    ;
0000                cpmsegment      equ     0       ; base of CP/M-86
0000                ccpoffset       equ     0000    ; offset of CCP
6000                bdosioffset     equ     6000h   ;
0B00                bdosoffset      equ     0b00h   ; offset of BDOS-86
2500                biosoffset      equ     2500h   ; offset of BIOS-86
1700                bdoscodesize    equ     1700h   ; size of BDOS code
2200                bdosdataseg     equ     bdosoffset+bdoscodesize
                    ;
```

CP/M 86 V.1.1 - Vol I
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # C81-162

```
 31
 32  =                  eject ! include bdosequ.a86     ; symbol definitions
 33  =                  ;*****************************************************************
 34  =                  ;                                                               *
 35  =                  ;                       BDOS symbols:                           *
 36  =                  ;                                                               *
 37  =                  ;*****************************************************************
 38  =                  ;
 39  =                  ;
 40  =  0022            dvers   EQU     22h                     ;version 2.2
 41  =                  ;
 42  =  FFFF            on      EQU     0ffffh
 43  =  0000            off     EQU     00000h
 44  =  FFFF            true    equ     0ffffh
 45  =  0000            false   equ     not true
 46  =  0000            testf   EQU     off
 47  =  0000            loader  equ     false
 48  =                  ;
 49  =                  ;       Special 8086 symbols:
 50  =                  ;
 51  =  0000            b       equ     byte ptr 0
 52  =  0000            w       equ     word ptr 0
 53  =                  ;
 54  =                  ;       bios value defined at end of module
 55  =                  ;
 56  =  0050            ssize   EQU     80                      ;96 level stack
 57  =                  ;
 58  =                  ;
 59  =  0000            cseg    cpmsegment
 60  =                  ;
 61  =                  ;       CCP entry
 62  =                  ;
 63  =                          org     ccpoffset
 64  =                  ccpent:
 65  =                          dseg
 66  =                          org     ccpoffset+0ah
 67  =000A              comlen  rs      1
 68  =                          org     ccpoffset+0bh
 69  =                  ccpbuf:
 70  =                          org     ccpoffset+805h
 71  =0005              submit  rb      1
 72  =                  ;
 73  =                  ;    BDOS equates
 74  =  0005            list_file       equ     5               ;printer function
 75  =  000D            diskio  equ     13              ;diskio function low value
 76  =                  ;
 77  =                  ;
 78  =                  ;
 79  =                  ;       BIOS routines
 80  =                  ;
 81  =                          org     biosoffset
 82  =                  bios:
 83  =  2500            bootf   EQU     BIOS+3*0        ;cold boot function
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        ;warm boot function
                        ;console status function
                        ;console input function
                        ;console output function
                        ;list output function
                        ;punch output function
                        ;reader input function
                        ;disk home function
                        ;select disk function
                        ;set track function
                        ;set sector function
                        ;set dma function
                        ;read disk function
                        ;write disk function
                        ;list status function
                        ;sector translate
                        ;set base for disk I/O
                        ;get segment table address
                        ;get I/O mapping byte
                        ;seg I/O mapping byte
                        ;number of BIOS functions
                        ;
                        ;       equates for non graphic characters
                                                ;control c
                                                ;physical eol
                                                ;backspace
                                                ;prnt toggle
                                                ;repeat line
                                                ;stop/start screen
                                                ;line delete
                                                ;=ctl-u
                                                ;end of file
                                                ;char delete
                                                ;tab char
                                                ;carriage return
                                                ;line feed
                                                ;up arrow
                        ;
                        ;
                        ;       module addresses
                        ;
                        ;       literal constants
                                                ;end of directory
                        ;
                        ;
                        ;       file control block (fcb) constants
                                                ;empty directory entry
                                                ;last record# in extent
                                                ;record size
                                                ;file control block size
                                                ;directory elts / record
                                                ;log2(dirrec)
```

[illegible] 1981, 1982
DIGITAL RESEARCH
P. O. B[illegible] X 579
PACIFIC GROVE, CA 93950
SER. #

```
                        
0003                    dskmsk  EQU     dirrec-1
0005                    fcbshf  EQU     5                       ;log2(fcblen)
                        ;
000C                    extnum  EQU     12                      ;extent number field
001F                    maxext  EQU     31                      ;largest extent number
000D                    ubytes  EQU     13                      ;unfilled bytes field
000E                    modnum  EQU     14                      ;data module number
000F                    maxmod  EQU     15                      ;largest module number
0080                    fwfmsk  EQU     80h                     ;file write flag is high
                                                                ;order modnum
000F                    namlen  EQU     15                      ;name length
000F                    reccnt  EQU     15                      ;record count field
0010                    dskmap  EQU     16                      ;disk map field
001F                    lstfcb  EQU     fcblen-1
0020                    nxtrec  EQU     fcblen
0021                    ranrec  EQU     nxtrec+1                ;random record field (2 bytes)
                        ;
                        ;       reserved file indicators
0007                    rofile  EQU     9                       ;high order of first type char
000A                    invis   EQU     10                      ;invisible file in dir command
                        ;       equ     11                      ;reserved
0060                    zero_readro     equ     060h            ;indicates open r/o user 0
00FE                    not_found       equ     0feh
0008                    f8      equ     byte ptr 8
0080                    interfc_bit     equ     80h             ;high bit for interface flag
007F                    not_interfc_bit equ     07fh            ;mask to zero high bit
                        ;
                        ;       Memory management symbols:
                        ;
0000                    membase         equ     word ptr 0
0002                    memlg           equ     word ptr 2
0004                    memdescrlg      equ     4
                        ;
                        ;       interrupt control indicators
                        ;
00E0                    bdosint equ     224             ;BDOS reserved interrupt number
0200                    interruptbit    equ     200h            ; bit 9 of flag word
                        ;
                        ;
                        ;
                        ;********************** end of BDOS symbols ************************
                        end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        eject ! include bdosent.a86    ; entry module
                        ;********************* BDOS entry module **********************
                        ;
                        ;       Perform branching, error handling and special
                        ;       8086 stuff.
                        ;
                        ;
                        ;
                        ;
                                if      not loader
                                org     0a60h
0A60 3132206A616E       db      '12 january 32'
     756172792038
     32
                                endif
                        ;
                        ;
                        ;
0000                    cseg    cpmsegment
                        org     bdosoffset-80h          ;dispatch table moved to offset A80h
                                                        ;for release 1.1
                        ;       dispatch table for functions
                        ;
0A80 260FA8156314       functab dw      func0,func1,func2,func3
     B015
0A88 1225IC15B515               dw      punchf,tolistf,func6,func7
     D315
0A90 D815DE15EA15               dw      func8,func9,func10,func11
     F115
000C                    diskf   equ     12                      ;beginning of disk functions
0A98 EC0A06209D1F               dw      func12,func13,func14,func15
     2720
0AA0 A120A720D220               dw      func16,func17,func18,func19
     EB20
0AA8 FA20FD200321               dw      func20,func21,func22,func23
     0C21
0AB0 15211B212121               dw      func24,func25,func26,func27
     2821
0AB8 BA172E213421               dw      func28,func29,func30,func31
     3D21
0AC0 462158216121               dw      func32,func33,func34,func35
     6721
0AC8 6D217321F715               dw      func36,func37,func38,func39
     F715
0AD0 CA21                       dw      func40
0AD2 D621                       dw      func47
0AD4 F821                       dw      func48
0AD6 F10A                       dw      func49                  ;vers 1.1 function
0AD8 570F760F800F               dw      func50,func51,func52,func53     ;8086 functions
     AC0F
0AE0 CE0FF40F0610               dw      func54,func55,func56,func57
     4410
0AE8 A4101D0E                   dw      func58,func59
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        ;
0029                    nfuncs80        equ     41      ;number of 8080 functions
002F                    sfunc86         equ     47      ;start number for 8086 functions
0036                    nfuncs          equ     54      ;total number of BDOS functions
                        ;
                        ;
                        func12:
0AEC B022                       mov     al,dvers
0AEE E9030B     15F4            jmp     staret
                        ;
                        ;
                        func49:
0AF1 BBA024                     mov     bx,offset dmaad
0AF4 E94A16     2141            jmp     sthlret
                        ;
                        ;
                        ;
                        ;
                        ;
                        org     bdosoffset
                        ;
0B00 000000000000               db      0,0,0,0,0,0
                        ;
                        ;       enter here from the user's program with function number in cl,
                        ;       and information address in dx
                        ;
                        ;
                        ;
                        ;
                        bdose:                          ;arrive here from user progs
0B06 1E                         push    ds              ;save user's DS
0B07 8CC8                       mov     ax,cs
0B09 8ED8                       mov     ds,ax           ;set BDOS DS
0B0B 8F069D24                   pop     parametersegment        ;user's DS register
                        ;
                                                        ;switch stack segment
0B0F 80F905                     cmp     cl,list_file
0B12 7405       0B19            jz      in_bdos         ;if listf....only one user in BDOS
                        ;
0B14 80F90D                     cmp     cl,diskio
0B17 7205       0B1E            jb      save_stacks     ;console....i/o...allow despool in BDOS
                        in_bdos:
0B19 C606B124FF                 mov     mxdos,true
                        ;
                        ;
                        save_stacks:
0B1E 8C169424                   mov     SS_save,ss
0B22 89269624                   mov     SP_save,sp
0B26 803E982401                 cmp     stack_flag,01
0B2B 7407       0B34            jz      bdose0
0B2D 8CC8                       mov     ax,cs
0B2F 8ED0                       mov     ss,ax
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
286
287 =0B31 BC3E24                    mov     sp,offset endstack
288 =                       ;
289 =                       bdose0:
290 =0B34 3B1E9624                  mov     bx,SP_save      ;enable interrupt if it was on
291 =0B38 8E069424                  mov     ES,SS_save      ;upon entry
292 =0B3C 268B5F04                  mov     bx,ES:4[bx]
293 =0B40 81E30002                  and     bx,interruptbit
294 =0B44 7401          0B47        jz      bdose1
295 =0B46 FB                        sti
296 =                       bdose1:                         ;pick up parameter segment
297 =0B47 8CD8                      mov     ax,ds
298 =                       ;
299 =0B49 55                        push    bp              ;save registers
300 =0B4A 57                        push    di
301 =0B4B 56                        push    si
302 =                       ;
303 =0B4C 0E                        push    cs              ;switch to local segment registers
304 =0B4D 07                        pop     es
305 =0B4E 8C062523                  mov     retrn_es,es     ;return BDOS data base..most calls
306 =                       ;
307 =                       ;
308 =0B52 E81A00        0B6F        call    oldbdos         ;call 8080 BDOS
309 =                       ;
310 =0B55 5E                        pop     si              ;unsave 8086 registers
311 =0B56 5F                        pop     di
312 =0B57 5D                        pop     bp
313 =0B58 8E062523                  mov     es,retrn_es         ;return data base in es   by ks
314 =                       ;
315 =                                                       ;switch back to entry stack
316 =0B5C FA                        cli
317 =0B5D 8E169424                  mov     ss,SS_save
318 =0B61 8B269624                  mov     sp,SP_save
319 =0B65 C606B12400                mov     mxdos,false     ;flag off on exit..no one in BDOS
320 =0B6A 8E1E9D24                  mov     ds,parametersegment
321 =0B6E CF                        iret                    ; return from interrupt
322 =                       ;
323 =                       ;
324 =                       oldbdos:                            ;8080 bdos entry point
325 =0B6F 89169924                  MOV     info,dx             ;info=dx
326 =0B73 88161023                  MOV     linfo,dl            ;linfo = low(info) - don't equ
327 =0B77 32C0                      XOR     AL,AL
328 =0B79 A39B24                    MOV     aret,ax             ;return value defaults to 0000
329 =0B7C A22723                    mov     parcopfl,al         ;set true if parameter
330 =                                                           ;block copied from user
331 =0B7F A21523                    MOV     fcbdsk,AL
332 =0B82 A21323                    MOV     resel,AL            ;fcbdsk,resel=false
333 =0B85 BB8921                    MOV     BX,offset goback    ;common return point
334 =0B88 53                        PUSH    BX                  ;jmp goback equivalent to ret
335 =0B89 80F929                    CMP     cl,nfuncs80
336 =0B8C 7209          0B97        jb      branch
337 =0B8E 80E906                    sub     cl,sfunc86-nfuncs80
338 =0B91 80F936                    cmp     cl,nfuncs
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. Box 579
PACIFIC GROVE, CA 93950
SER. #

```
                                
=0B94 7201          0B97        jb      branch
=0B96 C3                        RET
=                       ;                               ;skip if invalid #
=0B97 8AD9              branch: MOV     bl,cl
=0B99 B700                      mov     bh,0
=0B9B 8ACA                      MOV     CL,DL           ;possible output char. to CL
=0B9D 03DB                      add     bx,bx           ;branch
=0B9F 2EFFA7800A                jmp     cs:functab[bx]
=                       ;
=                       ;
=                       ;       error subroutines
=                       persub:                         ;report permanent error
=0BA4 BBAB23                    mov     BX,offset permsg
=0BA7 E81900        0BC3        CALL    errflg          ;to report the error
=0BAA 3C03                      CMP     AL,ctlc
=0BAC 7555          0C03        jnz     parret          ;rnz
=0BAE EB10          0BC0        jmps    jmp_f0free
=                                                       ;reboot if response is ctlc
=                                                       ;and ignore the error
=                       ;
=                       selsub:                         ;report select error
=0BB0 B3B623                    mov     BX,offset selmsg
=0BB3 EB08          0BBD        JMPS    waiterr         ;wait console before boot
=                       ;
=                       rodsub:                         ;report write to R/O disk
=0BB5 BBC223                    mov     BX,offset rodmsg
=0BB8 EB03          0BBD        JMPS    waiterr
=                       ;
=                       rofsub:                         ;report read/only file
=0BBA BBBD23                    mov     BX,offset rofmsg ;wait for console
=                       ;
=                       waiterr:
=                                                       ;wait for response before boot
=0BBD E80300        0BC3        CALL    errflg
=                       jmp_f0free:
=0BC0 E976A03       0F2D        JMP     f0free
=                       ;
=                       errflg:
=                                                       ;report error to console,
=                                                       ;message address in BX
=0BC3 53                        PUSH    BX
=0BC4 E8D308        149A        CALL    crlf            ;stack mssg address, new line
=0BC7 A0A424                    MOV     AL,curdsk
=0BCA 0441                      ADD     AL,'A'
=0BCC A2A723                    MOV     dskerr,AL       ;current disk name
=0BCF B79B23                    mov     CX,offset dskmsg
=0BD2 E8CF08        14A4        CALL    print           ;the error message
=0BD5 59                        POP     CX
=0BD6 E8CB08        14A4        CALL    print           ;error message tail
=0BD9 E9AA07        1386        jmp     conin           ;jump to input character
=                       ;
=                       parsave:
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
PACIFIC GROVE, CA 93950
SER. #

```
392
393 =                             ;copy parameterblock from user segment to BDOS segment
394 =                             ;CL-reg = lenght of parameter block
395 =                       ;
396 =0BDC 1E                       push    ds
397 =0BDD 50                       push    ax
398 =0BDE 51                       push    cx
399 =0BDF C6062723FF               mov     parcopfl,true
400 =0BE4 880E2823                 mov     parlg,cl
401 =0BE8 32ED                     xor     ch,ch
402 =0BEA 8B369924                 mov     si,info
403 =0BEE 8936A924                 mov     infosave,si
404 =0BF2 BFC623                   mov     di,offset loc_par_area
405 =0BF5 893E9924                 mov     info,di
406 =0BF9 8E1E9D24                 mov     ds,parametersegment
407 =0BFD FC                       cld
408 =0BFE F3A4              rep    movs    al,al
409 =0C00 59                       pop     cx
410 =0C01 58                       pop     ax
411 =0C02 1F                       pop     ds
412 =0C03 C3                parret: ret
413 =                       ;
414 =                       parsave33:
415 =                             ;copy 33 byte length parameterblock
416 =0C04 51                       push    cx
417 =0C05 B121                     mov     cl,33
418 =0C07 EB03          0C0C       jmps    pscommon
419 =                       ;
420 =                       parsave36:
421 =                             ;copy 36 byte length parameterblock
422 =0C09 51                       push    cx
423 =0C0A B124                     mov     cl,36
424 =                       pscommon:
425 =0C0C E8CDFF        0BDC       call    parsave
426 =0C0F 59                       pop     cx
427 =0C10 C3                       ret
428 =                       ;
429 =                       parunsave:
430 =                             ;copy local parameter block to user segment
431 =                       ;
432 =0C11 06                       push    es
433 =0C12 50                       push    ax
434 =0C13 51                       push    cx
435 =0C14 8A0E2823                 mov     cl,parlg
436 =0C18 32ED                     xor     ch,ch
437 =0C1A 8E069D24                 mov     es,parametersegment
438 =0C1E BEC623                   mov     si,offset loc_par_area
439 =0C21 8B3EA924                 mov     di,infosave
440 =0C25 893E9924                 mov     info,di
441 =0C29 FC                       cld
442 =0C2A F3A4              rep    movs    al,al
443 =0C2C 59                       pop     cx
444 =0C2D 58                       pop     ax
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
445
446  =0C2E 07                    pop     es
447  =0C2F C3                    ret
448  =                       ;
449  =                       ;
450  =                       ;****************** end BDOS entry module *****************
451                          end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        eject ! include bdosld.a86     ; new bdos load function
                        ;
                        ;     memory management function numbers .
                        ;
0035                    get_max_mem     equ     53      ;get maximum memory
0036                    get_max_abs     equu    54      ;maximum memory at abs base
0037                    alloc_mem       equ     55      ;allocate an amount of memory
0038                    alloc_mem_abs   equ     56      ;allocate at abs base
                         ;
                        ;   other function numbers
001A                    set_dmafn       equ     26      ;set dma address
0033                    set_dma_base    equ     51      ;set dma segment base address
                        ;
                        ;
                        ;
                        ;         bdos load program
                        ;
                        ;
                        ;
                        ;  absolute header equate symbols
                        ;
0000                    typseg  equ     byte ptr 0
0001                    actual  equ     word ptr 1
0003                    abs     equ     word ptr 3
0005                    min     equ     word ptr 5
0007                    max     equ     word ptr 7
                        ;
                        ;   page zero equates
                        ;
0000                    len     equ     word ptr 0
0002                    lenh    equ     byte ptr 2      ;high byte of max memory for segment
0003                    sbase   equ     word ptr 3
0005                    useripO equ     byte ptr 5      ;user's IP register
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;     use bx for totmin and dx for totmax
                        ;
                        ;     use bx for totmin and dx for totmax
                        ;
                        ;
                        readbuf:
0C30 B114                       mov     cl,14h
0C32 8B16DC22                   mov     dx,hold_info            ;fcb address in dx
0C36 8E1E2323                   mov     ds,user_parm_seg        ;user's segment
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
505
506  =0C3A E80500       0C42        call    shell_to_bdos
507  =0C3D 8CC8                     mov     ax,cs                   ;restore bdos DS
508  =0C3F 8ED8                     mov     ds,ax
509  =0C41 C3                       ret
510  =                          ;
511  =                          shell_to_bdos:
512  =                          ;
513  =0C42 2EC606982401             mov     cs:stack_flag,1    ;BDOS not to initialize stack
514  =0C48 CDE0                     int     bdosint            ;use main entry to BDOS
515  =0C4A C3                       ret
516  =                          ;
517  =                          ;
518  =                          ;
519  =                          get_memory:
520  =0C4B 53                       push    bx
521  =0C4C BAD722                   mov     dx,offset umembase
522  =0C4F E8F0FF       0C42        call    shell_to_bdos
523  =0C52 5B                       pop     bx
524  =0C53 C3                       ret
525  =                          ;
526  =                          get_min_first:                  ;change pointers (SI,DI) if
527  =0C54 50                       push    ax                  ;(SI) is not 'smallest'
528  =0C55 53                       push    bx
529  =0C56 8B5C07                   mov     bx,max[si]
530  =0C59 2B5C05                   sub     bx,min[si]
531  =0C5C 8B4507                   mov     ax,max[di]
532  =0C5F 2B4505                   sub     ax,min[di]
533  =0C62 3BD8                     cmp     bx,ax
534  =0C64 7602         0C68        jbe     g_m_f_1
535  =0C66 87F7                     xchg    si,di
536  =                          g_m_f_1:
537  =0C68 5B                       pop     bx
538  =0C69 58                       pop     ax
539  =0C6A C3                       ret
540  =                          ;
541  =                          ;
542  =                          getumem:
543  =0C6B 2BD8                     sub     bx,bx
544  =0C6D 2BD2                     sub     dx,dx
545  =0C6F C606E222FF               mov     mod8080,true
546  =                          ;
547  =0C74 B90800                   mov     cx,8                ;check 8 possible segments
548  =0C77 BE0C22                   mov     si,offset header
549  =                          seglp:
550  =0C7A 803C00                   cmp     typseg[si],0
551  =0C7D 7466         0CE5        jz      nxtseg
552  =0C7F 803C02                   cmp     typseg[si],2
553  =0C82 7505         0C89        jnz     notds
554  =0C84 C606E22200               mov     mod8080,false   ;this is not model 8080
555  =                          notds:
556  =0C89 837C0300                 cmp     word ptr abs[si],0h
557  =0C8D 743E         0CCD        jz      notabs
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
558
559 =0CBF 8B4403                  mov     ax,abs[si]
560 =0C92 A3D722                  mov     umembase,ax     ;header data to memory control block
561 =0C95 837C0700                cmp     max[si],0h
562 =0C99 8B4405                  mov     ax,min[si]
563 =0C9C 7403          0CA1      jz      strlen
564 =0C9E 8B4407                  mov     ax,max[si]
565 =                       strlen:
566 =0CA1 A3D922                  mov     umemlg,ax
567 =0CA4 51                      push    cx
568 =0CA5 B138                    mov     cl,alloc_mem_abs
569 =0CA7 52                      push    dx
570 =0CA8 E8A0FF        0C4B      call    get_memory
571 =0CAB 5A                      pop     dx
572 =0CAC 59                      pop     cx
573 =0CAD 803E^B2400              cmp     lret,00h
574 =0CB2 7431          0CE5      jz      nxtseg          ;successful allocate
575 =                                                     ;cannot get absolute segment
576 =0CB4 51                      push    cx
577 =0CB5 B136                    mov     cl,get_max_abs
578 =0CB7 E891FF        0C4B      call    get_memory      ;get maximum at absolute address
579 =0CBA 59                      pop     cx
580 =0CBB A1D922                  mov     ax,umemlg       ;returned maximum
581 =0CBE 394405                  cmp     min[si],ax
582 =0CC1 731D          0CE0      jnb     jmperr
583 =0CC3 894407                  mov     max[si],ax
584 =0CC6 8138                    mov     cl,alloc_mem_abs
585 =0CC8 E880FF        0C4B      call    get_memory      ;now allocate this maximum
586 =0CCB EB18          0CE5      jmps    nxtseg
587 =                       notabs:
588 =0CCD 035C05                  add     bx,min[si]
589 =0CD0 837C0700                cmp     max[si],0h      ;additional buffer space requested?
590 =0CD4 740C          0CE2      jz      nomax
591 =0CD6 035407                  add     dx,max[si]      ;user maximum for this group
592 =0CD9 730A          0CE5      jnc     nxtseg          ;check on oveflow
593 =0CDB BAF0FF                  mov     dx,0fff0h       ;change from 000 request
594 =0CDE EB05          0CE5      jmps    nxtseg
595 =                       ;
596 =                       jmperr:
597 =0CE0 EB5B          0D3D      jmps     retaff
598 =                       ;
599 =                       nomax:
600 =0CE2 033405                  add     dx,min[si]
601 =                       nxtseg:
602 =0CE5 83C609                  add     si,9
603 =0CE8 E290          0C7A      loop    segld
604 =0CEA 08D2                    or      dx,dx           ;only an absolute request
605 =0CEC 7429          0D17      jz      getu_mem_ok
606 =0CEE 8916D922                mov     umemlg,dx
607 =0CF2 B137                    mov     cl,alloc_mem    ;get maximum contained in dx
608 =0CF4 E854FF        0C4B      call    get_memory
609 =0CF7 803E9B2400              cmp     lret,0h
610 =0CFC 7419          0D17      jz      getumemok       ;system allocated maximum request
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
611
612 =0CFE B135                    mov     cl,get_max_mem  ;try maximum avail for min request
613 =0D00 E848FF        0C4B      call    get_memory
614 =0D03 3B1ED922                cmp     bx,umemlg       ;check with minimum requirement
615 =0D07 77D7          0CE0      ja      jmperr
616 =0D09 B137                    mov     cl,alloc_mem    ;allocate available memory
617 =0D0B E83DFF        0C4B      call    get_memory
618 =0D0E 8B16D922                mov     dx,umemlg       ;get additional space in DX
619 =0D12 2BD3                    sub     dx,bx
620 =0D14 E82C00        0D43      call    divmem          ;divide between data,extra or code
621 =                         getu_mem_ok:
622 =0D17 8B3E9423                mov     di,mem_stk_ptr
623 =0D1B A12323                  mov     ax,user_parm_seg
624 =0D1E 8CC9                    mov     cx,cs
625 =0D20 3BC1                    cmp     ax,cx
626 =0D22 7508          0D2C      jnz     notccp          ;call is not from OS
627 =0D24 C6450401                mov     byte ptr ccpflag [di],1
628 =0D28 FE069323                inc     nccpalloc       ;number of ccp memory allocations
629 =                         notccp:
630 =0D2C EB12          0D40      jmps    reta0
631 =                         ;
632 =                         findseg:
633 =0D2E BE0C22                  mov     si,offset header
634 =0D31 B90800                  mov     cx,8
635 =                         findseg1:
636 =0D34 3804                    cmp     typseg[si],al
637 =0D36 7408          0D40      jz      reta0       ;found the requested seg in header
638 =0D38 83C609                  add     si,9
639 =0D3B E2F7          0D34      loop    findseg1
640 =                         retaff:
641 =0D3D B0FF                    mov     al,0ffh
642 =0D3F C3                      ret
643 =                         reta0:
644 =0D40 B000                    mov     al,0
645 =0D42 C3                      ret
646 =                         ;
647 =                         ;
648 =                         divmem:
649 =0D43 803EE222FF              cmp     mod8080,true
650 =0D48 750B          0D55      jnz     divmem1
651 =0D4A 8B1E1122                mov     bx,min8080
652 =0D4E 03DA                    add     bx,dx           ;add the extra to code min and
653 =                                                     ; store in max
654 =0D50 891E1322                mov     max8080,bx
655 =0D54 C3                      ret
656 =                         divmem1:
657 =0D55 B002                    mov     al,2
658 =0D57 E8D4FF        0D2E      call    findseg
659 =0D5A 837C0700                cmp     max[si],0h
660 =0D5E 742E          0D8E      jz      alles
661 =                                             ;give extra to data segment if extra
662 =0D60 8BFE                    mov     di,si
663 =0D62 B80300                  mov     ax,3
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
664
665 =0D65 E8C6FF      0D2E        call    findseg
666 =0D68 837C0700              cmp     max[si],0h
667 =0D6C 7419          0D87        jz      allds
668 =0D6E E8EBFE        0C54        call    get_min_first   ; make shure that the 'smallest'
669 =                               ; segment comes first
670 =0D71 D1EA                  shr     dx,1    ;give 1/2 to data and 1/2 to extra
671 =0D73 8BDA                  mov     bx,dx
672 =0D75 035C05                add     bx,min[si]
673 =0D78 3B5C07                cmp     bx,max[si]      ; be shure that 1/2 is not too much
674 =0D7B 7607          0D84        jbe     allds1
675 =0D7D 2B5C07                sub     bx,max[si]
676 =0D80 03D3                  add     dx,bx
677 =0D82 EB03          0D87        jmps    allds
678 =0D84 895C07            allds1: mov     max[si],bx
679 =                       allds:
680 =0D87 035505                add     dx,min[di]
681 =0D8A 895507                mov     max[di],dx      ;all for data if extra was blank
682 =0D8D C3                    ret
683 =                       alles:
684 =0D8E B003                  mov     al,3
685 =0D90 E89BFF        0D2E        call    findseg
686 =0D93 035405                add     dx,min[si]
687 =0D96 895407                mov     max[si],dx
688 =0D99 C3                    ret
689 =                       ;
690 =                       ;
691 =                       ;
692 =                       ;
693 =                       nxtbase:
694 =                                               ;calculate base of next segment
695 =0D9A 837C0700              cmp     max[si],0
696 =0D9E 7404          0DA4        jz      nxtbas1
697 =0DA0 034407                add     ax,max[si]
698 =0DA3 C3                    ret
699 =                       nxtbas1:
700 =0DA4 034405                add     ax,min[si]
701 =0DA7 C3                    ret
702 =                       ;
703 =                       ;
704 =                       setbase0:
705 =                       ;                       calculate base addr. for independent segments
706 =                       ;                       store in absolute field of header
707 =                       ;
708 =                       ;
709 =0DAB C606D62200            mov     ip0,0h  ;assume model is not 8080 and code begins
710 =                                               ;at 0h
711 =                       ;
712 =0DAD A1D722                mov     ax,umembase
713 =0DB0 803EE222FF            cmp     mod8080,true
714 =0DB5 750D          0DC4        jnz     setbasea
715 =0DB7 C606D62201            mov     ip0,01h ;code begins at 100h relative to cs
716 =0DBC A30F22                mov     abs8080,ax
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
717
718 =0DBF A3D422                    mov     pag0,ax
719 =0DC2 EB2C          0DF0        jmps    set_base_end
720 =                           setbasea:
721 =0DC4 BE0C22                    mov     si,offset header
722 =0DC7 B90800                    mov     cx,8
723 =                           setbaseb:
724 =0DCA 803C00                    cmp     typseg[si],0
725 =0DCD 7419          0DE8        jz      setbased
726 =0DCF 803C02                    cmp     typseg[si],2
727 =0DD2 7503          0DD7        jnz     setbasec
728 =0DD4 A3D422                    mov     pag0,ax
729 =                           setbasec:
730 =0DD7 837C0300                  cmp     abs[si],0h
731 =0DDB 750E          0DE8        jnz     setbasee
732 =0DDD 894403                    mov     abs[si],ax
733 =0DE0 803C09                    cmp     typseg[si],9
734 =0DE3 7503          0DE8        jnz     setbased
735 =0DE5 C60401                    mov     typseg[si],1        ;change for MP/M
736 =                           setbased:
737 =0DE8 E8AFFF        0D9A        call    nxtbase
738 =                           setbasee:
739 =0DEB 83C609                    add     si,9    ;next segment
740 =0DEE E2DA          0DCA        loop    setbaseb
741 =                           set_base_end:
742 =0DF0 C3                        ret
743 =                           get16:
744 =0DF1 83FD70                    cmp     bp,112          ;end of input record
745 =0DF4 750E          0E04        jnz     get16a
746 =0DF6 53                        push    bx
747 =0DF7 06                        push    es
748 =0DF8 E835FE        0C30        call    readbuf
749 =0DFB 07                        pop     es
750 =0DFC 5B                        pop     bx              ;restore bx
751 =0DFD BD0000                    mov     bp,0
752 =0E00 BE5422                    mov     si,offset ccpbuff
753 =0E03 C3                        ret
754 =                           get16a:
755 =0E04 83C510                    add     bp,16
756 =0E07 C3                        ret
757 =                           ;
758 =                           slength:
759 =                                               ;slenght converts 16 bit address to 20 bit
760 =0E08 8BD8                      mov     bx,ax           ;save length which has 16 byte base
761 =0E0A 2500F0                    and     ax,0f000h       ;determine high 4 bits of address
762 =0E0D B104                      mov     cl,4
763 =0E0F D2EC                      shr     ah,cl
764 =                                                       ;ah has high order 4 bits rightshifted
765 =0E11 B104                      mov     cl,4
766 =0E13 D3E3                      shl     bx,cl           ;lengh is now 20 bit length
767 =0E15 83EB01                    sub     bx,1            ;set borrow bit if 0fffh
768 =0E18 7302          0E1C        jae     slen1
769 =0E1A FECC                      dec     ah
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        slen1:
=0E1C C3                        ret
                        ;
                        ;

                        ;
                        func59:
                        ;
=0E1D A19D24                    mov     ax,parametersegment
=0E20 A32323                    mov     user_parm_seg,ax        ;save during load calls to BDOS
=0E23 A19924                    mov     ax,info         ;save info for calls to BDOS
=0E26 A3DC22                    mov     hold_info,ax
=0E29 A19424                    mov     ax,SS_save
=0E2C A3E022                    mov     hold_sssave,ax          ;save transient stack
=0E2F A19624                    mov     ax,SP_save
=0E32 A3DE22                    mov     hold_spsave,ax
=0E35 8CCA                      mov     dx,cs           ;set base for dma transfer
=0E37 B133                      mov     cl,set_dma_base ;set base for dma transfer
=0E39 E806FE    0C42            call    shell_to_bdos
=0E3C BA0C22                    mov     dx,offset header ;header of memory image
=0E3F B11A                      mov     cl,set_dmafn
=0E41 E8FEFD    0C42            call    shell_to_bdos   ;call BIOS for setdma
=0E44 E8E9FD    0C30            CALL    readbuf         ;read the header
=0E47 803E9B2400                cmp     lret,00h        ;successful read?
=0E4C 7403      0E51            jz      load1
                        load_err_jmp:
=0E4E E9CD00    0F1E            jmp     load_err
                        ;
                        load1:
=0E51 E817FE    0C6B            call    getumem         ;get memory for user program
=0E54 84C0                      test      al,al
=0E56 75F6      0E4E            jnz     load_err_jmp
                        load2:
=0E58 E84DFF    0DA8            call    setbase0        ;distribute memory to various segments
                        ;
                        ;
=0E5B BA5422                    mov     dx,offset ccpbuff
=0E5E B11A                      mov     cl,set_dmafn
=0E60 E8DFFD    0C42            call    shell_to_bdos   ;set dma to bios
=0E63 BB7000                    mov     bp,112  ;force first read
=0E66 B90800                    mov     cx,8    ;load 8 segments
=0E69 BA0C22                    mov     bx,offset header ;header contains loading data
                        loadn0:
=0E6C 51                        push    cx
=0E6D 803F00                    cmp     typseg[bx],0

=0E70 7424      0E96            jz      nexload         ;segment not used
=0E72 8E4703                    mov     es,abs[bx]      ;use extra segment for destination
=0E75 BF0000                    mov     di,0
                        loadn1:
=0E78 8B4F01                    mov     cx,actual[bx]   ;absolute length of segment in file
=0E7B E319      0E96            jcxz    nexload
```

CCP/M-86™ - 1981, 1982
DIGITAL RESEARCH
P.O. B[illegible]
SER #_____

```
 823
 824 =                      loadn2:
 825 =0E7D 51                         push    cx              ;save loop control
 826 =0E7E E870FF       0DF1          call    get16           ;length is no. of 16 byte units
 827 =0E81 B90800                     mov     cx,8
 828 =0E84 FC                         cld
 829 =0E85 F3A5              rep      movs    ax,ax
 830 =0E87 83FF00                     cmp     di,0            ;moved more than 64k?
 831 =0E8A 7507         0E93          jnz     loadn3
 832 =0E8C 8CC2                       mov     dx,es
 833 =0E8E 80C610                     add     dh,10h  ;move to next 64K memory area
 834 =0E91 8EC2                       mov     es,dx
 835 =                      loadn3:
 836 =0E93 59                         pop     cx              ;loop control for the 16 bytes
 837 =0E94 E2E7         0E7D          loop    loadn2          ;end of this segment load??
 838 =                      ;
 839 =                      nexload:
 840 =0E96 59                         pop     cx
 841 =0E97 83C309                     add     bx,9            ;next segment in header
 842 =0E9A E2D0         0E6C          loop    loadn0          ;check end of all segments
 843 =                      ;
 844 =                                                              ;
 845 =                      ;
 846 =0E9C 8E06D422                   mov     es,pag0         ;move header data to user page0
 847 =0EA0 8C067B24                   mov     aret,es         ;save in aret for user
 848 =0EA4 BF0000                     mov     di,0
 849 =0EA7 B93000                     mov     cx,48
 850 =0EAA B000                       mov     al,0            ;zero first 48 byte in page zero
 851 =0EAC FC                         cld
 852 =0EAD F3AA              rep      stos    al
 853 =                      ;
 854 =0EAF BF0000                     mov     di,0            ;again zero di
 855 =0EB2 A0D622                     mov     al,ip0
 856 =0EB5 26884505                   mov     es:userip0[di],al       ;set value for initial IP load
 857 =0EB9 B201                       mov     dl,1            ;move descriptor for 8 segments
 858 =                      ;
 859 =                      ;
 860 =                      ;
 861 =                      filpag1:
 862 =0EBB 8E0C22                     mov     si,offset header ;base and length to user area
 863 =0EBE B90800                     mov     cx,8
 864 =                      filpag2:
 865 =0EC1 3814                       cmp     typseg[si],dl           ;segments in order
 866 =0EC3 7407         0ECC          jz      filpag3
 867 =0EC5 83C609                     add     si,9
 868 =0EC8 E2F7         0EC1          loop    filpag2
 869 =0ECA EB30         0EFC          jmps    filpag5
 870 =                      filpag3:
 871 =0ECC 8B4403                     mov     ax,abs[si]      ;store base
 872 =0ECF 26894503                   mov     es:sbase[di],ax
 873 =0ED3 8B4405                     mov     ax,min[si]
 874 =0ED6 837C0700                   cmp     max[si],0h
 875 =0EDA 7403         0EDF          jz      filpag4
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # _____

```
876
877 =0EDC 8B4407                  mov     ax,max[si]      ;use max if indicated
878 =                       filpag4:
879 =0EDF E826FF        0E08    call    slength         ;change to 20 bit addr, using 3 bytes
880 =0EE2 26886502              mov     es:lenh[di],ah          ;high byte of max memory
881 =0EE6 26891D                mov     es:len[di],bx   ;low 16 bit address
882 =0EE9 803EE222FF            cmp     mod8080,true
883 =0EEE 750C          0EFC    jnz     filpag5
884 =0EF0 83C706                add     di,6
885 =0EF3 26886502              mov     es:lenh[di],ah  ;high byte of max memory
886 =0EF7 26891D                mov     es:len[di],bx   ;low 16 bit address
887 =0EFA EB0A          0F06    jmps    load_ok
888 =                       filpag5:
889 =0EFC FEC2                  inc     dl
890 =0EFE 83C706                add     di,6
891 =0F01 80FA08                cmp     dl,8
892 =0F04 76B5          0EBB    jbe     filpag1
893 =                       ;
894 =                       load_ok:
895 =0F06 A1DC22                mov     ax,hold_info
896 =0F09 A39924                mov     info,ax         ;restore info
897 =0F0C C606982400            mov     stack_flag,0    ;clear stack flag on exit
898 =0F11 A1E022                mov     ax,hold_sssave  ;restore the real user's stack
899 =0F14 A39424                mov     SS_save,ax
900 =0F17 A1DE22                mov     ax,hold_spsave
901 =0F1A A39624                mov     SP_save,ax
902 =0F1D C3                    ret
903 =                       load_err:
904 =0F1E C7069B24FFFF          mov     aret,0ffffh
905 =0F24 EBE0          0F06    jmps     load_ok
906 =                       ;
907 =                       ;
908 =                       ;
909
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
910
911  =                        eject ! include bdosnew.a86    ; new 8086 BDOS functions
912  =                        ;
913  =                        ;******************** New BDOS functions for 8086 ****************
914  =                        ;
915  =                        func0:
916  =0F26 A01023                     mov     al,linfo
917  =0F29 3C01                       cmp     al,01h          ;01 indicates memory should not be freed
918  =0F2B 741F          0F4C         jz      restore_stack
919  =                        ;
920  =                        ;
921  =                        f0free:
922  =0F2D C6060A0000                 mov     byte ptr comlen,0h      ;zero the command length in CCP
923  =                        f0freea:
924  =0F32 803E932301                 cmp     nccpalloc,1     ;any entries in memory alloc stack
925  =0F37 7705          0F3E         ja      f0free1
926  =0F39 E86801        10A4         call    func58
927  =0F3C EB0E          0F4C         jmps     restore_stack
928  =                        ;
929  =                        ;
930  =                        f0free1:
931  =0F3E E8F201        1133         call    freeuser
932  =0F41 8B1E9423                   mov     bx,mem_stk_ptr
933  =0F45 E8E900        1031         call    free
934  =0F48 FE0E9323                   dec     nccpalloc
935  =                        ;
936  =                        restore_stack:
937  =0F4C C606982400                 mov     stack_flag,0            ;restore for normal entry
938  =0F51 BB0325                     mov     bx,offset wbootf
939  =0F54 E9B706        160E         jmp     thebios
940  =                        ;
941  =                        ;
942  =                        func50:
943  =                                ;call BIOS function number
944  =0F57 1E                         push    ds              ;pick up register from par. block
945  =0F58 8E1E9D24                   mov     ds,parametersegment
946  =0F5C 8BDA                       mov     bx,dx
947  =0F5E 8A07                       mov     al,[bx]         ;BIOS function no.
948  =0F60 8B4F01                     mov     cx,1[bx]        ;CX-reg
949  =0F63 8B5703                     mov     dx,3[bx]        ;DX-reg
950  =0F66 1F                         pop     ds
951  =0F67 5B                         pop     bx              ;skip normal return
952  =                        bioscont:
953  =0F68 33DB                       xor     bx,bx           ;branch to BIOS
954  =0F6A 8AD8                       mov     bl,al
955  =0F6C 02C0                       add     al,al           ;*2
956  =0F6E 02D8                       add     bl,al           ;*3
957  =0F70 81C30025                   add     bx,biosoffset
958  =0F74 FFE3                       jmp     bx
959  =                        ;
960  =                        ;
961  =                        func51:
962  =                                ;set segment base for disk I/O
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 963
 964 =                          ;
 965 =0F76 8916A224                     mov     dmabase,dx
 966 =0F7A 8BCA                         mov     cx,dx
 967 =0F7C E8B415        2533           call    setdmbf
 968 =0F7F C3                           ret
 969 =                          ;
 970 =                          ;
 971 =                          func52:
 972 =                                  ;get current dma base and offset
 973 =                                  ;returns double word
 974 =0F80 8B1EA224                     mov     bx,dmabase
 975 =0F84 891E2523                     mov     retrn_es,bx     ;return dmabase in ES
 976 =0F88 8B1EA024                     mov     bx,dmaad
 977 =0F8C E93211        2141           jmp     sthlret
 978 =                          ;
 979 =                          ;
 980 =                          ;
 981 =                          ;
 982 =                          ;       used by memory management routines for stack management
 983 =                          ;
 984 =                          ;       BX addresses user memory descriptor block
 985 =                          ;
 986 =                          ;
 987 =                          new_allocation:
 988 =0F8F 832E942305                   sub     mem_stk_ptr,5           ;update stk ptr
 989 =0F94 8B3E9423                     mov     di,mem_stk_ptr
 990 =0F98 8B07                         mov     ax,membase[bx]
 991 =0F9A 8905                         mov     stbase[di],ax
 992 =0F9C 8B4702                       mov     ax,memlg[bx]
 993 =0F9F 894502                       mov     stlen[di],ax
 994 =0FA2 C6450400                     mov     ccpflag[di],0           ; zero the ccp load flag
 995 =0FA6 0006292301                   add     nallocmem,1             ;no of entries in memory stack
 996 =0FAB C3                           ret
 997 =                          ;
 998 =                          func53:
 999 =                                  ;get maximum contiguous memory----do not allocate
1000 =0FAC E89B01        114A          call    get_mem_entry           ;prepare for loop
1001 =                          func53a:                               ;alt entry for load
1002 =0FAF 750A          0FBB          jnz     jp_mem_err
1003 =                          func53b:
1004 =0FB1 803E2A2300                  cmp     ncrmem,0                ;do we have any space available
1005 =0FB6 7473          102B          jz      get_mem_err
1006 =0FB8 E8ED01        11A8          call    get_max_seg             ;get maximum available
1007 =                          jp_mem_err:
1008 =0FBB 756E          102B          jnz     get_mem_err
1009 =0FBD 8B4402                      mov     ax,memlg[si]            ;return base and length
1010 =0FC0 394702                      cmp     memlg[bx],ax            ;check if greater than request
1011 =0FC3 7203          0FC8          jb      fun53b
1012 =0FC5 894702                      mov     memlg[bx],ax
1013 =                          fun53b:
1014 =0FC8 8B04                        mov     ax,membase[si]
1015 =0FCA 8907                        mov     membase[bx],ax
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER #

```
1016
1017 =0FCC EB53        1021        jmps    get_mem_ok
1018 =
1019 =                         func54:
1020 =                                 ;get maximum memory at specified base--no allocation
1021 =0FCE E87901      114A        call    get_mem_entry           ;prepare for loop
1022 =                         func54a:
1023 =0FD1 7558        102B        jnz     get_mem_err
1024 =                         func54b:
1025 =0FD3 83C604                  add     si,memdescrlg           ;move to next table elem.
1026 =0FD6 FEC1                    inc     cl
1027 =0FD8 3A0E2A23                cmp     cl,ncrmem               ;test if whole table searched
1028 =0FDC 734D        102B        jae     get_mem_err
1029 =                         ;
1030 =0FDE E84802      1229        call    test_within             ;test if user memory within
1031 =                                                             ;table element
1032 =0FE1 75F0        0FD3        jnz     func54b                  ;try next if not
1033 =0FE3 8B4402                  mov     ax,memlg[si]            ;return length of region
1034 =0FE6 0304                    add     ax,membase[si]          ;calculate end of region
1035 =0FE8 2B07                    sub     ax,membase[bx]          ;calculate length of absolute region
1036 =0FEA 394702                  cmp     memlg[bx],ax            ;check if > request
1037 =0FED 7203        0FF2        jb      fun54c
1038 =0FEF 894702                  mov     memlg[bx],ax
1039 =                         fun54c:
1040 =0FF2 EB2D        1021        jmps    get_mem_ok
1041 =                         ;
1042 =                         ;
1043 =                         func55:
1044 =                                 ;allocate an amount of memory
1045 =0FF4 E85301      114A        call    get_mem_entry           ;prepare for loop
1046 =                         func55a:
1047 =0FF7 7532        102B        jnz     get_mem_err             ;jump if error
1048 =                                                                 ;alt entry for load
1049 =0FF9 E8DD01      11D9        call    get_min_seg             ;loop for min. seg to fit
1050 =0FFC 752D        102B        jnz     get_mem_err             ;not found
1051 =0FFE E86B01      116C        call    alloc_seg       ;allocate new segment
1052 =1001 E88BFF      0F8F        call    new_allocation  ;push on memory allocation stack
1053 =1004 EB1B        1021        jmps    getmem_ok
1054 =                         ;
1055 =                         ;
1056 =                         func56:
1057 =                                 ;allocate an amout of memory at a specific segment base
1058 =1006 E84101      114A        call    get_mem_entry           ;prepare for loop
1059 =                         func56a:                            ;alt entry for load
1060 =1009 7520        102B        jnz     get_mem_err
1061 =                         getmal:
1062 =100B 83C604                  add     si,memdescrlg           ;move to next table elem.
1063 =100E FEC1                    inc     cl
1064 =1010 3A0E2A23                cmp     cl,ncrmem               ;test if whole table searched
1065 =1014 7315        102B        jae     get_mem_err
1066 =                         ;
1067 =1016 E81002      1229        call    test_within             ;test if user memory within
1068 =                                                             ;table element
```

COPYRIGHT : 1981, 1982
DIGITAL RESEARCH
P. O. Box 579
PACIFIC GROVE, CA 93950
SER. #

```
1069
1070  =1019 75F0        100B          jnz     getmal                  ;try next if not
1071  =101B E84302      1261          call    alloc_mem_at            ;allocate this segment
1072  =101E E86EFF      0F8F          call    new_allocation          ;push this allocation on stack
1073  =                           ;
1074  =                           get_mem_ok:
1075  =1021 A02A23                      mov     al,ncrmem
1076  =1024 884704                      mov     memdescrlg[bx],al
1077  =                           get_mem_ok1:
1078  =1027 32C0                        xor     al,al
1079  =1029 EB02        102D            jmps    get_mem_ex
1080  =                           get_mem_err:
1081  =102B B0FF                        mov     al,0ffh
1082  =                           get_mem_ex:
1083  =102D A27B24                      mov     lret,al
1084  =1030 C3                          ret
1085  =                           ;
1086  =                           free:
1087  =1031 8B4702                      mov     ax,stlen[bx]            ;save in temp UMR for free
1088  =1034 A3D922                      mov     umemlg,ax
1089  =1037 8B07                        mov     ax,stbase[bx]           ;move base address from stack
1090  =1039 A3D722                      mov     umembase,ax
1091  =103C BBD722                      mov     bx,offset umembase      ;change BX for free
1092  =103F E81101      1153          call    get_mem_entry1
1093  =1042 EB0E        1052          jmps    free01
1094  =                           ;
1095  =                           func57:
1096  =                           ;
1097  =                                 ;free an amount of memory at a specific segment base
1098  =                                 ; or user program  requested memory
1099  =1044 E80301      114A          call    get_mem_entry           ;prepare entry
1100  =1047 807F04FF                    cmp     byte ptr memdescrlg[bx],0ffh
1101  =104B 7505        1052          jnz     free01
1102  =104D E8E300      1133          call    freeuser
1103  =1050 EBCF        1021          jmps    get_mem_ok
1104  =                           free01:
1105  =1052 837F0200                    cmp     memlg[bx],0             ;zero length illegal
1106  =1056 74D3        102B          jz      get_mem_err
1107  =                                                                 ;test if memory physical
1108  =                                                                 ;available
1109  =1058 E86C02      12C7          call    test_within_physical
1110  =105B 75CE        102B          jnz     get_mem_err             ;if not, error
1111  =105D E89202      12F2          call    test_within_avail   ;test if memory is available
1112  =1060 74C9        102B          jz      get_mem_err             ;if so, error
1113  =                           ;
1114  =1062 E86300      10C8          call    adjust_stack
1115  =1065 3C00                      cmp     al,0            ;check if previously allocated
1116  =1067 75C2        102B          jnz     get_mem_err
1117  =                           ;
1118  =                           ;
1119  =1069 E8C402      1330          call    test_at_seg_end
1120  =106C 7516        1084          jnz     free2
1121  =106E 8BFE                      mov     di,si                   ;save pointer to found elem.
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1122
1123  =1070 8AE9                    mov     ch,cl
1124  =                         ;
1125  =1072 E8E602      135B        call    test_at_seg_start
1126  =1075 7408        107F        jz      free1
1127  =                         ;
1128  =                         ;       Case 1, add length to already existing element
1129  =                         ;
1130  =1077 8B4702                  mov     ax,memlg[bx]
1131  =107A 014502                  add     memlg[di],ax
1132  =107D EBA2        1021        jmps    get_mem_ok
1133  =                         ;
1134  =                         ;       Case 2, user seg. in the middle of 2 table seg's
1135  =                         ;
1136  =107F E83802      12BA  free1:  call    join3_mem_segments      ;join all 3 segmentinto 1
1137  =1082 EB9D        1021        jmps     get_mem_ok
1138  =                         ;
1139  =1084 E8D402      135B  free2:  call    test_at_seg_start
1140  =1087 750C        1095        jnz     free3
1141  =                         ;
1142  =                         ;       Case 3, user segment immediately followed by table segment
1143  =                         ;
1144  =1089 8B07                  mov     ax,membase[bx]
1145  =108B 8904                  mov     membase[si],ax
1146  =108D 8B4702                mov     ax,memlg[bx]
1147  =1090 014402                add     memlg[si],ax
1148  =                       j_get_mem_ok:
1149  =1093 EB8C        1021        jmps    get_mem_ok
1150  =                         ;
1151  =                         ;       Case 4, user segment not adjacent to any table segments
1152  =                         ;
1153  =1095 E8B701      124F  free3:  call    new             ;prepare to make new element
1154  =1098 8B07                  mov     ax,membase[bx]
1155  =109A 8904                  mov     membase[si],ax
1156  =109C 8B4702                mov     ax,memlg[bx]
1157  =109F 894402                mov     memlg[si],ax
1158  =10A2 EBEF        1093        jmps    j_get_mem_ok
1159  =                         ;
1160  =                       func58:
1161  =                               ;free all physical memory
1162  =10A4 33C0                    xor     ax,ax
1163  =10A6 A22923                  mov     nallocmem,al            ;clear no. of allocated
1164  =                                                             ;memory segments
1165  =10A9 A29323                  mov     nccpalloc,al            ;clear ccp count
1166  =10AC B89323                  mov     ax,offset (memstack+40 ) ; initialize stack ptr
1167  =10AF A39423                  mov     mem_stk_ptr,ax                ;set memory stack pointer
1168  =10B2 E89114      2536        call    getsegtabf              ;BIOS call
1169  =10B5 8BF3                    mov     si,bx                   ;copy BIOS table to BDOS tab
1170  =10B7 8A0C                    mov     cl,[si]
1171  =10B9 D0E1                    shl     cl,1
1172  =10BB D0E1                    shl     cl,1
1173  =10BD FEC1                    inc     cl
1174  =10BF 32ED                    xor     ch,ch
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1175
1176  =10C1 BF2A23              mov     di,offset ncrmem        ;start of BDOS table
1177  =10C4 FC                  cld
1178  =10C5 F3A4          rep   movs    al,al
1179  =10C7 C3                  ret
1180  =                   ;
1181  =                   ;     Subroutines for memory management routines:
1182  =                   ;
1183  =                   ;
1184  =                   adjust_stack:
1185  =10C8 8A0E2923            mov     cl,nallocmem    ;number of current allocations
1186  =10CC B500                mov     ch,0
1187  =                                                 ;last stack entry
1188  =10CE 8F3E23              mov     di,offset (mem_stack+mem_stack_length-5)
1189  =                   adjust1:
1190  =10D1 8B07                mov     ax,membase[bx] ;current entry to free
1191  =10D3 3905                cmp     stbase[di],ax           ;compare with stack
1192  =10D5 7421        10F8    jz      remove_stack
1193  =10D7 7317        10F0    jnb      next_stack             ;not within this allocation
1194  =                   ;
1195  =                   ;     check if top of this allocated region
1196  =                   ;
1197  =10D9 8B15                mov     dx,stbase[di]           ;check if last addresses are =
1198  =10DB 035502              add     dx,stlen[di]            ;high address of allocated in DX
1199  =10DE 034702              add     ax,memlg[bx]            ;high address of free req in AX
1200  =10E1 3BC2                cmp     ax,dx                   ;compare last addresses
1201  =10E3 750B        10F0    jnz     next_stack
1202  =10E5 8B4502              mov     ax,stlen[di]            ;reduce size in stack
1203  =10E8 2B4702              sub     ax,memlg[bx]
1204  =10EB 894502              mov     stlen[di],ax            ;user freed top of region
1205  =10EE EB40        1130    jmps     end_adjust_stack       ;now return the memory
1206  =
1207  =                   next_stack:
1208  =10F0 83EF05              sub     di,5                    ;next entry in stack
1209  =10F3 E2DC        10D1    loop    adjust1
1210  =                   stack_err:
1211  =10F5 B0FF                mov     al,0ffh
1212  =10F7 C3                  ret                             ;not found on stack
1213  =                   ;
1214  =                   remove_stack:
1215  =10F8 8B4702              mov     ax,memlg[bx]            ;check of free < allocated
1216  =10FB 3B4502              cmp     ax,stlen[di]            ;compare user length with stack
1217  =10FE 7411        1111    jz      remove_stack1
1218  =1100 73F3        10F5    jnb     stack_err               ;cannot return region > stack
1219  =1102 0305                add     ax,stbase[di]           ;change base in stack
1220  =1104 8905                mov     stbase[di],ax
1221  =1106 8B4502              mov     ax,stlen[di]            ;change length in stack
1222  =1109 2B4702              sub     ax,memlg[bx]
1223  =110C 894502              mov     stlen[di],ax
1224  =110F EB1F        1130    jmps    end_adjust_stack
1225  =                   remove_stack1:
1226  =1111 8B45FB              mov     ax,stbase-5[di]         ;lower element in stack
1227  =1114 8905                mov     stbase[di],ax           ;pop stack
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
1228
1229  =1116 8B45FD              mov     ax,stlen-5[di]
1230  =1119 894502              mov     stlen[di],ax
1231  =111C 8A45FF              mov     al,ccpflag-5[di]
1232  =111F 884504              mov     ccpflag[di],al          ;end of entry move
1233  =1122 83EF05              sub     di,5
1234  =1125 E2D1        10F8    loop    remove_stack
1235  =1127 FE0E2923            dec     nallocmem               ;decrement no. allocations
1236  =112B 8306942305          add     mem_stk_ptr,5           ;change stack pointer
1237  =                     end_adjust_stack:
1238  =1130 B000                mov     al,00h                  ;indicate free request found in stack
1239  =1132 C3                  ret
1240  =                         ;
1241  =                         ;
1242  =                         ; Free all memory allocated by user program
1243  =                         ;
1244  =                         freeuser:
1245  =1133 803E292300          cmp     nallocmem,0
1246  =1138 740F        1149    jz      end_free_user
1247  =113A 8B1E9423            mov     bx,mem_stk_ptr
1248  =113E 807F0400            cmp     ccpflag[bx],0
1249  =1142 7505        1149    jnz     end_free_user
1250  =1144 E8EAFE      1031    call    free
1251  =1147 EBEA        1133    jmps    freeuser
1252  =                     end_free_user:
1253  =1149 C3                  ret
1254  =                         ;
1255  =                         get_mem_entry:
1256  =114A B105                mov     cl,5                    ;copy parameter block
1257  =114C E88DFA      0BDC    call    parsave
1258  =114F 8B1E9924            mov     bx,info                 ;address of user parameter
1259  =                     get_mem_entry1:
1260  =1153 B1FF                mov     cl,0ffh                 ;prepare counter
1261  =1155 BE2723              mov     si,offset (crmem-memdescrlg)    ;SI-reg = address of
1262  =                                                         ;BDOS memory table
1263  =1158 B0FF                mov     al,0ffh                 ;error return
1264  =115A 803E292308          cmp     nallocmem,8             ;test if too many
1265  =                                                         ;allocated memory segments
1266  =115F 7308        1169    jae     getmentex
1267  =1161 837F0200            cmp     memlg[bx],0             ;zero length illegal
1268  =1165 7402        1169    jz      getmentex
1269  =1167 FEC0                inc     al                      ;ok exit
1270  =                     getmentex:
1271  =1169 0AC0                or      al,al
1272  =116B C3                  ret
1273  =                         ;
1274  =                         ;
1275  =                         alloc_seg:
1276  =                                 ;allocate segment for 'GETMEMORY 'routine
1277  =                                 ;SI = available segment address
1278  =                                 ;BX = user parameter address
1279  =                                 ;CL = table segment no.
1280  =                         ;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1281
1282  =116C 51                      push    cx
1283  =116D 884402                  mov     ax,memlg[si]            ;test if same lenghts
1284  =1170 2B4702                  sub     ax,memlg[bx]
1285  =1173 7509          117E      jnz     alloc1
1286  =                       ;
1287  =                       ;       segments are equal, remove current table element
1288  =                       ;
1289  =1175 8B04                    mov     ax,membase[si]
1290  =1177 8907                    mov     membase[bx],ax
1291  =1179 E80B00        1187      call    remove_table_element
1292  =117C EB07          1185      jmps    allocexit
1293  =                       ;
1294  =                       alloc1:
1295  =117E 894402                  mov     memlg[si],ax            ;update length of tab el.
1296  =1181 0304                    add     ax,membase[si]
1297  =1183 8907                    mov     membase[bx],ax
1298  =                       allocexit:
1299  =                                                             ;increment allocated
1300  =                                                             ;memory segment counter
1301  =1185 59                      pop     cx
1302  =1186 C3                      ret
1303  =                       ;
1304  =                       ;
1305  =                       remove_table_element:
1306  =                               ;remove current table memory element from BDOS table
1307  =                               ;SI = address of current table element
1308  =                               ;CL = element no
1309  =                       ;
1310  =1187 515657                  push    cx ! push si ! push di
1311  =118A A02A23                  mov     al,ncrmem
1312  =118D FEC8                    dec     al
1313  =118F A22A23                  mov     ncrmem,al               ;decrement element counter
1314  =1192 2AC1                    sub     al,cl                   ;do not compress table if
1315  =1194 740E          11A4      jz      remex                   ;last element
1316  =                       ;
1317  =1196 8AC8                    mov     cl,al                   ;compress table
1318  =1198 32ED                    xor     ch,ch
1319  =119A 03C9                    add     cx,cx                   ;*2
1320  =119C 8BFE                    mov     di,si
1321  =119E 83C604                  add     si,memdescrlg
1322  =11A1 FC                      cld
1323  =11A2 F3A5          rep       movs    ax,ax
1324  =                       ;
1325  =11A4 5F5E59        remex:    pop     di ! pop si ! pop cx
1326  =11A7 C3                      ret
1327  =                       ;
1328  =                       get_max_seg:
1329  =                               ;compute maximum memory segment
1330  =                               ;CL = 0ffh
1331  =                               ;SI = start of segment table - 4
1332  =                               ;BX = address of user parameter
1333  =                       ;
```

CP/M 86 V.1.1 - VOL I
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER # CS1-162

```
1334
1335  =11A8 5257                    push dx ! push di
1336  =11AA 33D2                    xor     dx,dx                   ;preset max value
1337  =11AC B5FF                    mov     ch,0ffh                 ;flag
1338  =                       getmax1:
1339  =11AE 83C604                  add     si,memdescrlg
1340  =11B1 FEC1                    inc     cl
1341  =11B3 3A0E2A23                cmp     cl,ncrmem
1342  =11B7 730F        11C8        jae     getmaxex
1343  =                       ;
1344  =11B9 8B4402                  mov     ax,memlg[si]
1345  =11BC 3BC2                    cmp     ax,dx                   ;test against max value
1346  =11BE 72EE        11AE        jb      getmax1
1347  =11C0 8BD0                    mov     dx,ax                   ;new max value
1348  =11C2 8AE9                    mov     ch,cl
1349  =11C4 8BFE                    mov     di,si
1350  =11C6 EBE6        11AE        jmps    getmax1
1351  =                       ;
1352  =                       getmaxex:
1353  =11C8 B0FF                    mov     al,0ffh
1354  =11CA 8BF7                    mov     si,di
1355  =11CC 8ACD                    mov     cl,ch
1356  =11CE FEC5                    inc     ch
1357  =11D0 7402        11D4        jz      getmaxfin
1358  =11D2 FEC0                    inc     al
1359  =                       getmaxfin:
1360  =11D4 0AC0                    or      al,al
1361  =11D6 5F5A                    pop di  ! pop dx
1362  =11D8 C3                      ret
1363  =                       ;
1364  =                       ;
1365  =                       ;
1366  =                       get_min_seg:
1367  =                             ;compute minimum memory segments that user seg fits into
1368  =                             ;CL = 0ffh
1369  =                             ;SI = start of segment table - 4
1370  =                             ;BX = address of user parameter
1371  =                       ;
1372  =11D9 5257                    push dx ! push di
1373  =11DB BAFFFF                  mov     dx,0ffffh               ;preset min value
1374  =11DE B5FF                    mov     ch,0ffh                 ;flag
1375  =                       getmin1:
1376  =11E0 83C604                  add     si,memdescrlg
1377  =11E3 FEC1                    inc     cl
1378  =11E5 3A0E2A23                cmp     cl,ncrmem
1379  =11E9 7316        1201        jae     getminex
1380  =                       ;
1381  =11EB 8B4402                  mov     ax,memlg[si]
1382  =11EE 2B4702                  sub     ax,memlg[bx]
1383  =11F1 741A        120D        jz      getminfin
1384  =11F3 72EB        11E0        jb      getmin1                 ;to small table segment
1385  =11F5 3BC2                    cmp     ax,dx                   ;test against min value
1386  =11F7 73E7        11E0        jnb     getmin1
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1387
1388  =11F9 8BD0                    mov     dx,ax                   ;new min value
1389  =11FB 8AE9                    mov     ch,cl
1390  =11FD 8BFE                    mov     di,si
1391  =11FF EBDF        11E0        jmps    getmin1
1392  =                         ;
1393  :                         getminex:
1394  =1201 B0FF                    mov     al,0ffh
1395  =1203 8BF7                    mov     si,di
1396  =1205 8ACD                    mov     cl,ch
1397  =1207 FEC5                    inc     ch
1398  =1209 7402        120D        jz      getminfin
1399  =120B FEC0                    inc     al
1400  =                         getminfin:
1401  =120D 0AC0                    or      al,al
1402  =120F 5F5A                    pop di  ! pop dx
1403  =1211 C3                      ret
1404  =                         ;
1405  =                         ;
1406  =                         compute_end_seg:
1407  =                               ;compute segment address of end of memory segment
1408  =                               ;BX = meory segment pointer
1409  =1212 8B4702                    mov     ax,memlg[bx]
1410  =1215 0307                      add     ax,membase[bx]
1411  =1217 C3                        ret
1412  =                         ;
1413  =                         ;
1414  =                         compute_end_segments:
1415  =                               ;compute segment address of end of user memory and
1416  =                               ;table element
1417  =                               ;BX = user segment address
1418  =                               ;SI = table segment address
1419  =                               ;BP = address of result area in stack
1420  =                         ;
1421  =  0000                         tab_end_seg     equ     word ptr 0
1422  =  0002                         user_end_seg    equ     word ptr 2
1423  =                         ;
1424  =1218 53                        push    bx
1425  =1219 E8F6FF      1212          call    compute_end_seg
1426  =121C 894602                    mov     user_end_seg[bp],ax
1427  =121F 8BDE                      mov     bx,si
1428  =1221 E8EEFF      1212          call    compute_end_seg
1429  =1224 894600                    mov     tab_end_seg[bp],ax
1430  =1227 5B                        pop     bx
1431  =1228 C3                        ret
1432  =                         ;
1433  =                         ;
1434  =                         test_within:
1435  =                               ;test if user segment is within table segment
1436  =                               ;BX = address of user segment
1437  =                               ;SI = address of table segment
1438  =                         ;     tab_end_seg     equ     word ptr 0
1439  =                         ;     user_end_seg    equ     word ptr 2
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER #_____

```
1440
1441  =                     ;
1442  =1229 5583EC048BEC            push bp ! sub sp,4 ! mov bp,sp
1443  =122F 51                      push    cx
1444  =1230 B5FF                    mov     ch,0ffh                 ;error flag
1445  =1232 E8E3FF      1218       call    compute_end_segments
1446  =1235 3B07                    mov     ax,membase[bx]
1447  =1237 3B04                    cmp     ax,membase[si]
1448  =1239 720A        1245       jb      testifex                ;error
1449  =123B 8B4600                  mov     ax,tab_end_seg[bp]
1450  =123E 3B4602                  cmp     ax,user_end_seg[bp]
1451  =1241 7202        1245       jb      testifex
1452  =1243 B500                    mov     ch,0                    ;ok flag
1453  =                     testifex:
1454  =1245 8AC5                    mov     al,ch
1455  =1247 59                      pop     cx
1456  =1248 83C4045D                add sp,4  ! pop bp
1457  =124C 0AC0                    or      al,al
1458  =124E C3                      ret
1459  =                     ;
1460  =                     ;
1461  =                     new:
1462  =                             ;prepare for a new entry in BDOS memory table
1463  =124F A02A23                  mov     al,ncrmem
1464  =1252 FE062A23                inc     ncrmem
1465  =1256 B404                    mov     ah,memdescrlg
1466  =1258 F6E4                    mul     ah
1467  =125A 8BF0                    mov     si,ax
1468  =125C 8DB42B23                lea     si,crmem[si]
1469  =1260 C3                      ret
1470  =                     ;
1471  =                     ;
1472  =                     alloc_mem_at:
1473  =                             ;allocate memory at a specific base and with a given length
1474  =                             ;BX = address of user memory
1475  =                             ;SI = BDOS table element address
1476  =                     ;       tab_end_seg     equu    word ptr 0
1477  =                     ;       user_end_seg    equu    word ptr 2
1478  =                     ;
1479  =1261 5583EC048BEC            push bp ! sub sp,4 ! mov bp,sp
1480  =1267 5251                    push dx ! push cx
1481  =1269 E8ACFF      1218       call    compute_end_segments
1482  =126C 8B07                    mov     ax,membase[bx]
1483  =126E 3B04                    cmp     ax,membase[si]
1484  =1270 7519        128B       jnz     alla2
1485  =1272 8B4602                  mov     ax,user_end_seg[bp]
1486  =1275 3B4600                  cmp     ax,tab_end_seg[bp]
1487  =1278 7507        1281       jnz     alla1
1488  =                     ;
1489  =                     ;       Case 1, segments are equal.
1490  =                     ;
1491  =127A 59                      pop     cx                      ;restore table element no.
1492  =127B 51                      push    cx
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1493
1494  =127C E808FF      1187        call    remove_table_element
1495  =127F EB32        1283        jmps    allaex
1496  =                       ;
1497  =                       ;       Case 2, Segments have save start address, userseg < tabseg
1498  =                       ;
1499  =1281 8904              alla1:  mov     membase[si],ax
1500  =1283 8B4702                    mov     ax,memlg[bx]
1501  =1286 294402                    sub     memlg[si],ax
1502  =1289 EB28        12B3          jmps    allaex
1503  =                       ;
1504  =                       alla2:
1505  =128B 8B4600                    mov     ax,tab_end_seg[bp]
1506  =128E 2B4602                    sub     ax,user_end_seg[bp]
1507  =1291 7509        129C          jnz     alla3
1508  =                       ;
1509  =                       ;       Case 3, segments end at same address, userseg < tableseg
1510  =                       ;
1511  =1293 8B07                      mov     ax,membase[bx]
1512  =1295 2B04                      sub     ax,membase[si]
1513  =1297 894402                    mov     memlg[si],ax
1514  =129A EB17        12B3          jmps    allaex
1515  =                       ;
1516  =                       ;       Case 4, no boundaries are equal, userseg < tableseg
1517  =                       ;
1518  =129C 8B07              alla3:  mov     ax,membase[bx]          ;update table el. length
1519  =129E 2B04                      sub     ax,membase[si]
1520  =12A0 894402                    mov     memlg[si],ax
1521  =                       ;
1522  =12A3 E8A9FF      124F          call    new                     ;create new table element
1523  =12A6 8B4602                    mov     ax,user_end_seg[bp]
1524  =12A9 8904                      mov     membase[si],ax
1525  =12AB 2B4600                    sub     ax,tab_end_seg[bp]
1526  =12AE F7D8                      neg     ax
1527  =12B0 894402                    mov     memlg[si],ax
1528  =                       allaex:
1529  =12B3 595A                      pop cx ! pop dx
1530  =12B5 83C4045D                  add sp,4 ! pop bp
1531  =12B9 C3                        ret
1532  =                       ;
1533  =                       ;
1534  =                       join3_mem_segments:
1535  =                               ;join 2 table elements with user segment in the middle
1536  =                               ;DI = start table element address
1537  =                               ;SI = last table element address
1538  =                               ;BX = user element address
1539  =                               ;CL = last segment element no.
1540  =12BA 8B4702                    mov     ax,memlg[bx]
1541  =12BD 034402                    add     ax,memlg[si]
1542  =12C0 014502                    add     memlg[di],ax
1543  =12C3 E8C1FE      1187          call    remove_table_element
1544  =12C6 C3                        ret
1545  =                       ;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1546
1547  =                     test_within_physical:
1548  =                             ;test if user segment is within physical memory
1549  =                             ;BX = address of user segment
1550  =12C7 515256                  push cx ! push dx ! push si
1551  =12CA 53                      push    bx
1552  =12CB E86812      2536        call    getseqtabf          ;get BIOS segment table adr
1553  =12CE 8A2F                    mov     ch,[bx]             ;get no. of segments
1554  =12D0 8BF3                    mov     si,bx
1555  =12D2 83EE03                  sub     si,memdescrlg-1
1556  =12D5 5B                      pop     bx
1557  =12D6 B1FF                    mov     cl,0ffh             ;loop counter
1558  =                     physl:
1559  =12D8 83C604                  add     si,memdescrlg
1560  =12DB FEC1                    inc     cl
1561  =12DD 3ACD                    cmp     cl,ch
1562  =12DF 7309        12EA        jae     physerr
1563  =12E1 E845FF      1229        call    test_within         ;test if user segment is
1564  =                                                         ;within table segment
1565  =12E4 75F2        12D8        jnz     physl               ;if not, try next
1566  =12E6 B000                    mov     al,0                ;ok exit
1567  =12E8 EB02        12EC        jmps    physex
1568  =                     physerr:
1569  =12EA B0FF                    mov     al,0ffh
1570  =                     physex:
1571  =12EC 5E5A59                  pop si ! pop dx ! pop cx
1572  =12EF 0AC0                    or      al,al
1573  =12F1 C3                      ret
1574  =                     ;
1575  =                     ;
1576  =                     test_within_avail:
1577  =                             ;test if user segment is witnin any of the allocated
1578  =                             ;memory segments
1579  =                             ;BX = user segment
1580  =                     ;       tab_end_seg     equu    word ptr 0
1581  =                     ;       user_end_seg    equ     word ptr 2
1582  =                     ;
1583  =12F2 5583EC048BEC            push bp ! sub sp,4 ! mov bp,sp
1584  =12F8 5651                    push si ! push cx
1585  =12FA BE2723                  mov     si,offset (crmem-memdescrlg)
1586  =12FD B1FF                    mov     cl,0ffh
1587  =12FF 83C604          tav1:   add     si,memdescrlg
1588  =1302 FEC1                    inc     cl
1589  =1304 3A0E2A23                cmp     cl,ncrmem
1590  =1308 731B        1325        jae     tavnot
1591  =                     ;
1592  =130A E803FF      1218        call    compute_end_segments
1593  =130D 8B07                    mov     ax,membase[bx]
1594  =130F 3B04                    cmp     ax,membase[si]
1595  =1311 7309        131C        jnb     tav2
1596  =1313 8B4602                  mov     ax,user_end_seg[bp]
1597  =1316 3B04                    cmp     ax,membase[si]
1598  =1318 7707        1321        ja      tavfound
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1599
1600  =131A EBE3        12FF        jmps    tav1
1601  =                         tav2:
1602  =131C 3B4600                  cmp     ax,tab_end_seg[bp]
1603  =131F 730E        12FF        jae     tav1
1604  =                         tavfound:
1605  =1321 B000                    mov     al,0
1606  =1323 EB02        1327        jmps    tavex
1607  =                         tavnot:
1608  =1325 B0FF                    mov     al,0ffh
1609  =1327 595E                tavex:  pop cx ! pop si
1610  =1329 83C4045D                add sp,4 ! pop bp
1611  =132D 0AC0                    or      al,al
1612  =132F C3                      ret
1613  =                         ;
1614  =                         ;
1615  =                         test_at_seg_end:
1616  =                         ;
1617  =                         ;       tab_end_seg     equ     word ptr 0
1618  =                         ;       user_end_seg    equ     word ptr 2
1619  =                         ;
1620  =1330 5583EC048BEC                push bp ! sub sp,4 ! mov bp,sp
1621  =1336 BE2723                  mov     si,offset (crmem-memdescrlg)
1622  =1339 B1FF                    mov     cl,0ffh
1623  =                         lab1:
1624  =133B 83C604                  add     si,memdescrlg
1625  =133E FEC1                    inc     cl
1626  =1340 3A0E2A23                cmp     cl,ncrmem
1627  =1344 730C        1352        jae     labnot
1628  =                         ;
1629  =1346 E8CFFE      1218        call    compute_end_segments
1630  =1349 8B07                    mov     ax,membase[bx]
1631  =134B 2B4600                  sub     ax,tab_end_seg[bp]
1632  =134E 75EB        133B        jnz     lab1
1633  =1350 EB02        1354        jmps    labex
1634  =1352 B0FF                labnot: mov     al,0ffh
1635  =                         labex:
1636  =1354 83C4045D                add     sp,4 ! pop bp
1637  =1358 0AC0                    or      al,al
1638  =135A C3                      ret
1639  =                         ;
1640  =                         ;
1641  =                         test_at_seg_start:
1642  =                         ;
1643  =                         ;       tab_end_seg     equ     word ptr 0
1644  =                         ;       user_end_seg    equ     word ptr 2
1645  =                         ;
1646  =135B 5583EC048BEC                push bp ! sub sp,4 ! mov bp,sp
1647  =1361 BE2723                  mov     si,offset (crmem-memdescrlg)
1648  =1364 B1FF                    mov     cl,0ffh
1649  =                         llab1:
1650  =1366 83C604                  add     si,memdescrlg
1651  =1369 FEC1                    inc     cl
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1652
1653  =136B 3A0L2A23                cmp     cl,ncrmem
1654  =136F 730C          137D      jae     llabnot
1655  =                         ;
1656  =1371 E8A4FE        1218      call    compute_end_segments
1657  =1374 8B4602                  mov     ax,user_end_seg[bp]
1658  =1377 2B04                    sub     ax,membase[si]
1659  =1379 75EB          1366      jnz     llab1
1660  =137B EB02          137F      jmps    llabex
1661  =                         llabnot:
1662  =137D B0FF                    mov     al,0ffh
1663  =                         llabex:
1664  =137F 83C4045D                add sp,4 ! pop bp
1665  =1383 0AC0                    or      al,al
1666  =1385 C3                      ret
1667  =                         ;
1668  =                         ;
1669  =                         ;****************** End of new 8086 functions ******************
1670                            end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER #_____

```
1671
1672  =                    eject ! include bdosio.a86      ; byte I/O module
1673  =                    ;****************** BDOS byte I/O module ******************
1674  =                    ;
1675  =                    ;
1676  =                    ;       console handlers
1677  =                    conin:
1678  =                                                            ;read console character to A
1679  =1386 BBE622                 mov     bx,offset kbchar
1680  =1389 8A07                   MOV     AL,[bx]
1681  =138B C60700                 MOV     b[bx],0
1682  =138E 0AC0                   OR      AL,AL
1683  =1390 7525           13B7    jnz     ret0                    ;rnz
1684  =                                                            ;no previous char ready
1685  =1392 E96C02         1601    JMP     toconinf                  ;get character externally
1686  =                                                            ;ret
1687  =                    ;
1688  =                    conech:
1689  =                                                            ;read character with echo
1690  =1395 E8EEFF         1386    CALL    conin
1691  =1398 E80A00         13A5    CALL    echoc
1692  =139B 721A           13B7    JB      ret0                    ;rc
1693  =                                                            ;echo char before return
1694  =139D 50                     PUSH    AX
1695  =139E 8AC8                   MOV     CL,AL
1696  =13A0 E8C000         1463    CALL    tabout
1697  =13A3 58                     POP     AX
1698  =13A4 C3                     RET                             ;with character in A
1699  =                    ;
1700  =                    echoc:
1701  =                                                            ;echo character if graphic
1702  =                                                            ;cr, lf, tab, or backspace
1703  =13A5 3C0D                   CMP     AL,cr
1704  =13A7 740E           13B7    jz      ret0                    ;rz
1705  =13A9 3C0A                   CMP     AL,lf
1706  =13AB 740A           13B7    jz      ret0                    ;rz
1707  =13AD 3C09                   CMP     AL,tab
1708  =13AF 7406           13B7    jz      ret0                    ;rz
1709  =13B1 3C08                   CMP     AL,ctlh
1710  =13B3 7402           13B7    jz      ret0                    ;rz
1711  =13B5 3C20                   CMP     AL,' '
1712  =13B7 C3             ret0:   RET                             ;carry set if not graphic
1713  =                    ;
1714  =                    conbrk:                                 ;check for character ready
1715  =13B8 A0E622                 MOV     AL,kbchar
1716  =13BB 0AC0                   OR      AL,AL
1717  =13BD 754B           140A    JNZ     conb1                   ;skip if active kbchar
1718  =                                                            ;no active kbchar,
1719  =                                                            ;check external break
1720  =13BF E84402         1606    CALL    taconstf
1721  =13C2 2401                   AND     AL,1
1722  =13C4 74F1           13B7    jz      ret0                    ;rz
1723  =                                                            ;character ready, read it
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1724
1725 =13C6 E83802      1601        CALL    toconinf              ;to A
1726 =13C9 3C13                    CMP     AL,ctls
1727 =13CB 753A        1407        JNZ     conb0                 ;check stop screen function
1728 =                                                           ;found ctls, read next char
1729 =13CD E83102      1601        CALL    toconinf              ;to A
1730 =13D0 3C03                    CMP     AL,ctlc
1731 =13D2 7403        13D7        jz      ctlc_check
1732 =                                                           ;not a reboot,
1733 =                                                           ;act as if nothing happened
1734 =13D4 32C0                    XOR     AL,AL
1735 =13D6 C3                      RET                           ;with zero in accumulator
1736 =                         ;
1737 =                         ctlc_check:
1738 =                                 if      not loader
1739 =13D7 C606E52200                  mov     column,0h       ;restore for next CCP commar
1740 =13DC 803E932300                  cmp     nccpalloc,0
1741 =13E1 750B        13EE            jnz     ctlc_check0             ;reset disk and memory
1742 =13E3 E8200C      2006            call    func13          ;reset disk system
1743 =13E6 A09B24                      mov     al,lret         ;submit return flag
1744 =13E9 A20508                      mov     submit,al       ;let CCP know about submit
1745 =13EC EB0E        13FC            jmps    to_restore      ;jump to CCP
1746 =                                                         ;reenter current disk
1747 =                         ctlc_check0:
1748 =13EE F606AF24FF                  test    background,true ;check for despool in background
1749 =13F3 740F        1404            jz      ctlc_check1
1750 =13F5 803E932301                  cmp     nccpalloc,1     ;check for foreground to abort
1751 =13FA 7508        1404            jnz     ctlc_check1
1752 =                         to_restore:
1753 =13FC C6060A0000                  mov     byte ptr comlen,0h
1754 =                         ;
1755 =1401 E948FB      0F4C            jmp     restore_stack
1756 =                         ;
1757 =                                 endif
1758 =                         ;
1759 =                         ctlc_check1:
1760 =1404 E926FB      0F2D            jmp     f0free          ;process like func0 with 0 parm
1761 =                         ;
1762 =                         ;
1763 =                         conb0:
1764 =                                                         ;character in accum, save it
1765 =1407 A2E622                      MOV     kbchar,AL
1766 =                         conb1:
1767 =                                                         ;return with true set
1768 =140A B001                        MOV     AL,1
1769 =140C C3                          RET
1770 =                         ;
1771 =                         conout:
1772 =                                 ;compute character position/write conso
1773 =                                                         ;compcol = true if computing
1774 =                                                         ;column position
1775 =140D A0E322                      MOV     AL,compcol
1776 =1410 0AC0                        OR      AL,AL
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. B. X 5. 9
PACIFIC GROVE, CA 93950
SER. #

```
1777
1778 =1412 7516          142A          JNZ     compout
1779 =                                                      ;write the character,
1780 =                                                      ;then compute column
1781 =                                                      ;write console char from CL
1782 =1414 51                         PUSH    CX
1783 =1415 E8A0FF        1388          CALL    conbrk       ;check for screen stop func.
1784 =1418 59                         POP     CX
1785 =1419 51                         PUSH    CX           ;recall/save character
1786 =141A E  E01        160B          CALL    toconouf      ;externally, to console
1787 =141D 59                         POP     CX
1788 =141E 51                         PUSH    CX           ;recall/save character
1789 =                                                      ;may be copying to list dev.
1790 =141F 803EB62400                 cmp     listcp,0
1791 =1424 7403          1429          JZ      L_11
1792 =1426 EP 301        15FC          CALL    tolistf
1793 =                        L_11:                         ;to printer, if so
1794 =1429 59                         POP     CX           ;recall the character
1795 =                        compout:
1796 =142A 8AC1                       MOV     AL,CL        ;recall the character
1797 =                                                      ;and compute column position
1798 =142C BBE522                     mov     BX,offset column      ;A = char, BX = .column
1799 =142F 3C7F                       CMP     AL,rubout
1800 =1431 7416          1449          jz      ret1         ;no column change if nulls
1801 =1433 FE07                       INC     b[bx]        ;column = column + 1
1802 =1435 3C20                       CMP     AL,' '
1803 =1437 7310          1449          jae     ret1         ;rnc
1804 =                                                      ;not graphic, reset column
1805 =1439 FE0F                       DEC     b[bx]        ;column = column - 1
1806 =143B 8A07                       MOV     AL,[bx]
1807 =143D 0AC0                       OR      AL,AL
1808 =143F 7408          1449          jz      ret1         ;return if at zero
1809 =                                                      ;not at zero, may be
1810 =                                                      ;bacspace or end of line
1811 =1441 8AC1                       MOV     AL,CL        ;character back to A
1812 =1443 3C08                       CMP     al,ctlh
1813 =1445 7503          144A          JNZ     notbacksp
1814 =                                                      ;backspace character
1815 =1447 FE0F                       DEC     b[bx]        ;column = column - 1
1816 =1449 C3                 ret1:   RET
1817 =                        notbacksp:
1818 =                                                      ;not backspace, eol ?
1819 =144A 3C0A                       CMP     AL,lf
1820 =144C 75FB          1449          jnz     ret1         ;return if not
1821 =                                                      ;end of line, column = 0
1822 =144E C60700                     MOV     b[bx],0      ;column = 0
1823 =1451 C3                         RET
1824 =                        ;
1825 =                        ctlout:
1826 =                                ;send C character with possible preceding up-arrow
1827 =1452 8AC1                       MOV     AL,CL
1828 =1454 E84EFF        13A5          CALL    echoc        ;try if not graphic or spec.
1829 =1457 730A          1463          JAE     tabout       ;skip if graphic, tab,
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1830
1831  =                                                    ;cr,lf, or ctlh
1832  =                                                    ;send preceding up arrow
1833  =1459 50                    PUSH    AX
1834  =145A B15E                  MOV     CL,ctl
1835  =145C E8AEFF      140D      CALL    conout           ;up arrow
1836  =145F 59                    POP     cx
1837  =1460 80C940                OR      cl,40h           ;becomes graphic letter
1838  =                                                    ;ready to print
1839  =                                                    ;(drop through to tabout)
1840  =                    ;
1841  =                    tabout:
1842  =                                                    ;expand tabs to console
1843  =1463 80F909                CMP     cl,tab
1844  =1466 75A5        140D      jnz     conout           ;direct to conout if not
1845  =                                                    ;tab encountered, move to
1846  =                                                    ;next tab position
1847  =                    tab0:
1848  =1468 B120                  MOV     CL,' '
1849  =146A E8A0FF      140D      CALL    conout           ;another blank
1850  =146D A0E522                MOV     AL,column
1851  =1470 2407                  AND     AL,111b          ;column mod 8 = 0 ?
1852  =1472 75F4        1468      JNZ     tab0             ;back for another if not
1853  =1474 C3          ret2:     RET
1854  =                    ;
1855  =                    ;
1856  =                    backup:
1857  =                                                    ;back-up one screen position.
1858  =1475 E80500      147D      CALL    pctlh
1859  =1478 B120                  MOV     CL,' '
1860  =147A E88E01      160B      CALL    toconouf
1861  =                    ;       (drop through to pctlh)
1862  =                    pctlh:
1863  =                            ;send ctlh to console without affecting column count
1864  =147D B108                  MOV     CL,ctlh
1865  =147F E98901      160B      JMP     toconouf
1866  =                                                    ;ret
1867  =                    ;
1868  =                    crlfp:
1869  =                            ;print $, cr, lf for ctlx, ctlu, ctlr functions
1870  =                            ;then move to strtcol (starting column)
1871  =1482 B123                  MOV     CL,'$'
1872  =1484 E886FF      140D      CALL    conout
1873  =1487 E81000      149A      CALL    crlf
1874  =                                                    ;column = 0, move to
1875  =                                                    ;position strtcol
1876  =                    crlfp0:
1877  =148A A0L522                MOV     AL,column
1878  =148D 3A06E422              cmp     al,strtcol
1879  =1491 73E1        1474      jae     ret2             ;rnc
1880  =1493 B120                  MOV     CL,' '
1881  =1495 E875FF      140D      CALL    conout           ;print blank
1882  =1498 EBF0        148A      jmps    crlfp0
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
PACIFIC GROVE, CA 93950
SER. #

```
1883
1884 =                       ;
1885 =                       crlf:
1886 =                               ;carriage return line feed sequence
1887 =149A B10D                      MOV     CL,cr
1888 =149C E86EFF       140D         CALL    conout
1889 =149F B10A                      MOV     CL,lf
1890 =14A1 E969FF       140D         JMP     conout
1891 =                                                       ;ret
1892 =                       ;
1893 =                       print:
1894 =                               ;print message until M(BC) = '$'
1895 =14A4 8BF1                      MOV     SI,CX
1896 =14A6 AC                        LODS    al
1897 =14A7 3C24                      CMP     AL,'$'
1898 =14A9 74C9         1474         jz      ret2                    ;rz
1899 =                                                       ;more to print
1900 =14AB 41                        INC     CX
1901 =14AC 51                        PUSH    CX
1902 =14AD 8AC8                      MOV     CL,AL                   ;char to C
1903 =14AF 1E                        push    ds                      ;restore data segment
1904 =14B0 16                        push    ss
1905 =14B1 1F                        pop     ds
1906 =14B2 E8AEFF       1463         CALL    tabout                  ;another character printed
1907 =14B5 1F                        pop     ds
1908 =14B6 59                        POP     CX
1909 =14B7 EBEB         14A4         jmps    print
1910 =                       ;
1911 =                       read:
1912 =                               ;read to info address (max length, current length, buffer)
1913 =14B9 A0E522                    MOV     AL,column
1914 =14BC A2E422                    MOV     strtcol,AL              ;save start for ctl-x, ctl-h
1915 =14BF 8B1E9724                  MOV     BX,info
1916 =14C3 268A0F                    MOV     CL,es:[bx]
1917 =14C6 43                        INC     BX
1918 =14C7 53                        PUSH    BX
1919 =14C8 B500                      MOV     CH,0
1920 =                                                       ;CH = current buffer length,
1921 =                                                       ;CL = maximum buffer length,
1922 =                                                       ;BX= next to fill - 1
1923 =                       readnx:
1924 =                               ;read next character, CX, BX active
1925 =14CA 51                        PUSH    CX
1926 =14CB 53                        PUSH    BX                      ;blen, cmax, BX saved
1927 =                       readn0:
1928 =14CC E8B7FE       1386         CALL    conin                   ;next char in AL
1929 =                       ;       AND     AL,7fh                  ;mask parity bit
1930 =14CF 5B                        POP     BX
1931 =14D0 59                        POP     CX                      ;reactivate counters
1932 =14D1 3C0D                      CMP     AL,cr
1933 =14D3 7503         14D8         JNZ     L_19
1934 =14D5 E9C600       159E eofre:  JMP     readen
1935 =                       L_19:                                   ;end of line?
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
1936
1937  =14D8 3C0A                CMP     AL,lf
1938  =14DA 74F9        14D5    jz      eofre           ;jump to end of read if ctlh
1939  =14DC 3C08                CMP     AL,ctlh
1940  =14DE 750E        14EE    JNZ     noth            ;backspace?
1941  =                                                 ;do we have any characters
1942  =                                                 ;to backspace ?
1943  =14E0 0AED                OR      ch,ch
1944  =14E2 74E6        14CA    JZ      readnx
1945  =                                                 ;characters remain in buffer,
1946  =                                                 ;backup one
1947  =14E4 FECD                DEC     CH              ;remove one character
1948  =14E6 A0E522              MOV     AL,column
1949  =14E9 A2E322              MOV     compcol,AL      ;col > 0
1950  =                                                 ;compcol > 0 marks repeat as
1951  =                                                 ;length compute
1952  =14EC EB51        153F    jmps    linelen         ;uses same code as repeat
1953  =                     noth:
1954  =                                                 ;not a backspace
1955  =14EE 3C7F                CMP     AL,rubout
1956  =14F0 750C        14FE    JNZ     notrub          ;rubout char?
1957  =                                                 ;rubout if possible
1958  =14F2 0AED                OR      ch,ch
1959  =14F4 74D4        14CA    JZ      readnx          ;skip if len=0
1960  =                                                 ;buffer has characters,
1961  =                                                 ;resend last char
1962  =14F6 268A07              MOV     AL,es:[bx]
1963  =14F9 FECD                DEC     CH
1964  =14FB 4B                  DEC     BX              ;al=last char
1965  =                                                 ;blen=blen-1,
1966  =                                                 ;next to fill - 1 decremented
1967  =14FC EB7C        157A    JMps     rdech1          ;act like this is an echo
1968  =                     ;
1969  =                     notrub:
1970  =                                                 ;not a rubout char, check eol
1971  =14FE 3C05                CMP     AL,ctle
1972  =1500 750C        150E    JNZ     note            ;physical end line?
1973  =                                                 ;yes, save active counters
1974  =                                                 ;and force eol
1975  =1502 51                  PUSH    CX
1976  =1503 53                  PUSH    BX
1977  =1504 E873FF      147A    CALL    crlf
1978  =1507 C606E42200          MOV     strtcol,0       ;start position = 00
1979  =                     to_readn0:
1980  =150C EBBC        14CC    jmps    readn0          ;for another character
1981  =                     note:
1982  =                                                 ;not end of line, list toggle?
1983  =150E 3C10                CMP     AL,ctlp
1984  =1510 7506        1518    JNZ     notp            ;skip if not ctlp
1985  =                                                 ;list toggle - change parity
1986  =1512 F616B624            not     listcp          ;listcp = not listcp
1987  =1516 EBB2        14CA    jmps    readnx          ;for another char
1988  =                     notp:
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1989
1990  =                                                    ;not a ctlp, line delete?
1991  =1518 3C18                CMP     AL,ctlx
1992  =151A 7514          1530  JNZ     notx
1993  =151C 5B                  POP     BX                ;discard start position
1994  =                                                    ;loop while column > strtcol
1995  =                    backx:
1996  =151D A0E422              MOV     AL,strtcol
1997  =1520 BBE522              mov     bx,offset column
1998  =1523 90                  nop
1999  =1524 3A07                cmp     al,[bx]
2000  =1526 7391          14B9  jnb     read
2001  =1528 90                  nop
2002  =1529 FE0F                DEC     b[bx]
2003  =152B E847FF        1475  CALL    backup            ;column = column - 1
2004  =152E EBED          151D  jmps    backx             ;one position
2005  =                    notx:
2006  =                                                    ;not a control x, control u?
2007  =                                                    ;not control-X, control-U?
2008  =1530 3C15                CMP     AL,ctlu
2009  =1532 7507          153B  JNZ     notu              ;skip if not
2010  =                                                    ;delete line (ctlu)
2011  =1534 E84BFF        1482  CALL    crlfp             ;physical eol
2012  =1537 5B                  POP     BX                ;discard starting position
2013  =1538 E97EFF        1489  JMp     read              ;to start all over
2014  =                    notu:
2015  =                                                    ;not line delete, repeat line?
2016  =153B 3C12                CMP     AL,ctlr
2017  =153D 7535          1574  JNZ     notr
2018  =                    linelen:
2019  =                                                    ;repeat line, or compute
2020  =                                                    ;line len (ctlh)
2021  =                                                    ;if compcol > 0
2022  =153F 51                  PUSH    CX
2023  =1540 E83FFF        1482  CALL    crlfp             ;save line length
2024  =1543 59                  POP     CX
2025  =1544 5B                  POP     BX
2026  =1545 53                  PUSH    BX
2027  =1546 51                  PUSH    CX
2028  =                                                    ;bcur, cmax active,
2029  =                                                    ;beginning at BX
2030  =                    rep0:
2031  =1547 0AED                or      ch,ch
2032  =1549 740F          155A  JZ      rep1              ;count len to 00
2033  =154B 43                  INC     BX
2034  =154C 268A0F              MOV     CL,es:[bx]          ;next to print
2035  =154F FECD                DEC     CH
2036  =1551 51                  PUSH    CX
2037  =1552 53                  PUSH    BX                ;count length down
2038  =1553 E8FCFE        1452  CALL    ctlout            ;character echoed
2039  =1556 5B                  POP     BX
2040  =1557 59                  POP     CX                ;recall remaining count
2041  =1558 EBED          1547  jmps    rep0              ;for the next character
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
2042
2043  =                       rep1:
2044  =
2045  =
2046  =155A 53                        PUSH    BX              ;save next to fill
2047  =155B A0E322                    MOV     AL,compcol
2048  =155E 0AC0                      OR      AL,AL           ;>0 if computing length
2049  =1560 7410          1572        jz      backsp1
2050  =
2051  =
2052  =1562 2A06E522                  SUB     AL,column       ;diff > 0
2053  =1566 A2E322                    MOV     compcol,AL      ;count down below
2054  =
2055  =                       backsp:
2056  =
2057  =1569 E809FF        1475       CALL    backup          ;one space
2058  =156C FE0EE322                  DEC     compcol
2059  =1570 75F7          1569       JNZ     backsp
2060  =                       backsp1:
2061  =1572 EB98          150C       jmps    to_readn0
2062  =                       notr:
2063  =
2064  =                       rdecho:
2065  =1574 43                        INC     BX
2066  =1575 268807                    MOV     es:[bx],al      ;character filled to mem
2067  =1578 FEC5                      INC     CH              ;blen = blen + 1
2068  =                       rdech1:
2069  =
2070  =157A 51                        PUSH    CX              ;look for a random contr. char.
2071  =157B 53                        PUSH    BX              ;active values saved
2072  =157C 8AC8                      MOV     CL,AL           ;ready to print
2073  =157E E8D1FE        1452       CALL    ctlout          ;may be up-arrow C
2074  =1581 5B                        POP     BX
2075  =1582 59                        POP     CX
2076  =1583 268A07                    MOV     AL,es:[bx]      ;recall char
2077  =1586 3C03                      CMP     AL,ctlc         ;set flags for reboot test
2078  =1588 8AC5                      MOV     AL,CH           ;move length to A
2079  =158A 750B          1597       JNZ     notc            ;skip if not a control c
2080  =158C 3C01                      CMP     AL,1            ;control C, must be length 1
2081  =158E 7507          1597       JNZ     L_23
2082  =1590 8CD8                      mov     ax,ds           ;restore the ES register
2083  =1592 8EC0                      mov     es,ax           ;to base of operating system
2084  =1594 E940FE        13D7       JMP     ctlc_check
2085  =                       L_23:                           ;reboot if blen = 1
2086  =                                                       ;length not 1, ignore rebout
2087  =                       notc:
2088  =                                                       ;not reboot, end of buffer?
2089  =1597 3AC1                      CMP     AL,CL
2090  =1599 7303          159E       JAE     L_24
2091  =1598 E972CF        14CA       JMP     readnx
2092  =                       L_24:                           ;go for another if not
2093  =                       readen:
2094  =                                                       ;end of read , store blen
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

```
2095
2096  =159E 5B                          POP     BX
2097  =159F 26882F                      MOV     es:[bx],ch              ;M(current len) = B
2098  =15A2 B10D                        MOV     CL,cr
2099  =15A4 8CD8                        mov     ax,ds                   ;restore the ES register
2100  =15A6 8EC0                        mov     es,ax                   ;to base of operating system
2101  =15A8 E962FE      140D           JMP     conout                  ;return carriage
2102  =                                                                  ;ret
2103  =                         func1:
2104  =                                ;return console character with echo
2105  =                         ;
2106  =15AB E8E7FD      1395           CALL    conech
2107  =15AE EB44        15F4           JMPS    staret
2108  =                         ;
2109  =  1463                  func2   EQU     tabout
2110  =                                ;write console character with tab expansion
2111  =                         ;
2112  =                         ;
2113  =                         func3:
2114  =                                ;return reader character
2115  =                         ;
2116  =15B0 E8620F      2515           CALL    readerf
2117  =15B3 EB3F        15F4           JMPS    staret
2118  =                         ;
2119  =                         ;func4: equated to punchf
2120  =                                ;write punch character
2121  =                         ;
2122  =                         ;func5: equated to listf
2123  =                                ;write list character
2124  =                                ;write to list device
2125  =                         ;
2126  =                         func6:
2127  =                                ;direct console i/o - read if 0ffh
2128  =                         ;
2129  =15B5 8AC1                        MOV     AL,CL
2130  =15B7 FEC0                        INC     AL
2131  =15B9 7409        15C4           JZ      dirinp                  ;0ffh => 00h, means input mode
2132  =15BB FEC0                        INC     al
2133  =15BD 754C        160B           jnz     toconouf
2134  =15BF E84400      1606           call    toconstf
2135  =15C2 EB30        15F4           jmps    staret
2136  =                         dirinp:
2137  =15C4 E83F00      1606           CALL    toconstf                ;status check
2138  =15C7 0AC0                        OR      AL,AL
2139  =15C9 7503        15CE           JNZ     L_26
2140  =15CB E9E60B      21B4           JMP     retmon
2141  =                         L_26:                                   ;skip, return 00 if not ready
2142  =                                                                 ;character is ready, get it
2143  =15CE E83300      1601           CALL    toconinf                ;to A
2144  =15D1 EB21        15F4           JMPS    staret
2145  =                         ;
2146  =                         func7:
2147  =                                ;return io byte
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
2148
2149  =                     ;
2150  =15D3 E8630F  2539    call    getiobf
2151  =15D6 EB1C    15F4    JMPS staret
2152  =                     ;
2153  =                     func8:
2154  =                             ;set i/o byte
2155  =                     ;
2156  =15D8 8BCA                    mov     cx,dx           ;iobyte in dx
2157  =15DA E85F0F  253C            call    setiobf
2158  =15DD C3                      RET                             ;jmp goback
2159  =                     ;
2160  =                     func9:
2161  =                             ;write line until $ encountered
2162  =                     ;
2163  =15DE 8BCA                    MOV     CX,DX                   ;CX=string addre
2164  =15E0 1E                      push    ds                      ;read from user segment
2165  =15E1 8E1E9D24                mov     ds,parametersegment
2166  =15E5 E8BCFE  14A4            call    print                   ;out to console
2167  =15E8 1F                      pop     ds
2168  =15E9 C3                      ret
2169  =                     ;
2170  =                     func10:
2171  =                             ;read a buffered console line
2172  =15EA 8E067D24                mov     es,parametersegment
2173  =15EE E9C8FE  14B9            jmp     read
2174  =                     ;
2175  =                     ;
2176  =                     func11:
2177  =                             ;check console status
2178  =                     ;
2179  =15F1 E8C4FD  13B8            CALL    conbrk
2180  =                                                             ;(drop through to sta$ret)
2181  =                     staret:
2182  =                                                             ;store the A register to aret
2183  =15F4 A27B24                  MOV     lret,al
2184  =                     funcret:                                ;
2185  =15F7 C3                      RET                             ;jmp goback
2186  =                                                             ;(pop stack for non CP/M func)
2187  =                     ;
2188  =                     setlret1:
2189  =                                                             ;set lret = 1
2190  =15F8 B001                    MOV     AL,1
2191  =15FA EBF8    15F4            jmps    staret
2192  =                     ;
2193  =                     ;
2194  =                     ;
2195  =                     tolistf:
2196  =15FC BB0425                  mov     bx,offset listf
2197  =15FF EB0D    160E            jmps    thebios
2198  =                     ;
2199  =                     toconinf:
2200  =1601 BB0925                  mov     bx,offset coninf        ;all console functions will
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER #_____

```
2201                                                    ;use common entry to BIOS
2202  =
2203  =1604 EB08        160E        jmps    thebios
2204  =                         ;
2205  =                         toconstf:
2206  =1606 BB0625                  mov     bx,offset constf    ;all console functions use this
2207  =1609 EB03        160E        jmps    thebios             ;function
2208  =                         ;
2209  =                         toconouf:
2210  =160C BB0025                  mov     bx,offset conouf
2211  =                         ;
2212  =                         thebios:
2213  =                             if      not loader
2214  =160E FF1E0022                callf   dword ptr biosptr
2215  =                             endif
2216  =                             if      loader
2217  =                             call    bx
2218  =                             endif
2219  =1612 C3                      ret
2220  =                         ;
2221  =                         ;
2222  =                         ;   actual entry to  BIOS console and list handlers
2223  =                         ;
2224  =1613 FFD3                biosc1: call    bx
2225  =1615 CB                      retf
2226  =                         ;
2227  =                         ;
2228  =                         ;       end of Basic I/O System
2229  =                         ;
2230  =                         ;*****************************************************************
2231                            end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER #_____

```
2232
2233 =                    eject ! include bdosf1.a86     ; file system part 1
2234 =                    ;
2235 =                    ;*****************************************************************
2236 =                    ;*****************************************************************
2237 =                    ;**                                                             **
2238 =                    ;**  B a s i c   D i s k   O p e r a t i n g   S y s t e m      **
2239 =                    ;**                                                             **
2240 =                    ;*****************************************************************
2241 =                    ;*****************************************************************
2242 =                    ;
2243 =                    ;
2244 =                    ;       fixed addresses referenced in bios module are
2245 =                    ;       pererr (0009), selerr (000c), roderr (000f)
2246 =                    ;
2247 =                    ;       error message handlers
2248 =                    ;
2249 =                    ;pererror:
2250 =                                                        ;report permanent error
2251 =                    ;       lxi h,pererr  ;jmp goerr
2252 =                    ;
2253 =                    ;roderror:
2254 =                                                        ;report read/only disk error
2255 =                    ;       lxi h,roderr   ;jmp goerr
2256 =                    ;
2257 =                    ;roferror:
2258 =                                                        ;report read/only file error
2259 =                    ;       lxi h,rofer     ;jmp goerr
2260 =                    ;
2261 =                    selerror:
2262 =                                                        ;report select error
2263 =1616 BB0622                   mov     BX,offset seler
2264 =                    ;
2265 =                    ;
2266 =                    goerr:
2267 =                                                        ;BX = .errorhandler,call subr.
2268 =1619 FF27                     JMP     w[bx]           ;to subroutine
2269 =                    ;
2270 =                    ;
2271 =                    ;
2272 =                    ;       local subroutines for bios interface
2273 =                    ;
2274 =                    move:
2275 =                            ;move data length of length CL from source DX to
2276 =                            ;destination given by BX
2277 =161B 51                     push    cx
2278 =161C B500                   mov     ch,0
2279 =161E 8BF2                   mov     si,dx
2280 =1620 8BFB                   mov     di,bx
2281 =1622 FC                     cld
2282 =1623 F3A4             rep   movs    al,al
2283 =1625 59                     pop     cx
2284 =1626 C3                     ret
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER #_____

```
2285
2286  =                     ;
2287  =                     ;
2288  =                     selectdisk:
2289  =                             ;select the disk drive given by curdsk, and fill
2290  =                             ;the base addresses curtrka - alloca, then fill
2291  =                             ;the values of the disk parameter block
2292  =                     ;
2293  =1627 8A0E4424                MOV     cl,curdsk               ;current disk# to CL
2294  =                                                             ;lsb of DL = 0 if not yet
2295  =                                                             ;logged in
2296  =162B E8ED0E      251B        CALL    seldskf                 ;BX filled by call
2297  =                                                             ;BX = 0000 if error,
2298  =                                                             ;otherwise disk headers
2299  =162E 83FB00                  cmp     bx,0
2300  =1631 7450        1683        jz      ret4                    ;rz
2301  =1633 8B17                    MOV     DX,[bx]
2302  =1635 83C302                  add     bx,2
2303  =1638 891EEC22                MOV     cdrmaxa,BX
2304  =163C 83C302                  add     bx,2
2305  =163F 891EEE22                MOV     curtrka,BX
2306  =1643 83C302                  add     bx,2
2307  =1646 891EF022                MOV     curreca,BX
2308  =164A 83C302                  add     bx,2
2309  =
2310  =164D 87DA                    xchg    bx,dx
2311  =164F 891E0923                MOV     tranv,bx                ;.tran vector
2312  =                     ;
2313  =1653 8B1EF222                mov     bx,buffa                ;save dir buff for context
2314  =1657 891EAD24                mov     cbuffa,bx
2315  =                     ;
2316  =165B BBF222                  mov     BX,offset buffa         ;DX= source for move, BX=dest
2317  =165E B108                    MOV     CL,addlist
2318  =1660 E8B8FF      161B        CALL    move                    ;addlist filled
2319  =                                                             ;now fill the disk
2320  =                                                             ;parameter block
2321  =1663 8B16F422                MOV     dx,dpbaddr
2322  =1667 BBFA22                  mov     BX,offset sectpt        ;BX is destination
2323  =166A B10F                    MOV     CL,dpblist
2324  =166C E8ACFF      161B        CALL    move                    ;data filled
2325  =                                                             ;set single/double map mode
2326  =166F A00023                  MOV     al,byte ptr maxall+1    ;largest allocation number
2327  =1672 BB1223                  mov     BX,offset single
2328  =1675 C607FF                  MOV     b[bx],true                      ;assume a=00
2329  =1678 0AC0                    OR      AL,AL
2330  =167A 7403        167F        JZ      retselect
2331  =                                                             ;high order of maxall not
2332  =                                                             ;zero, use double dm
2333  =167C C60700                  MOV     b[bx],false
2334  =                     retselect:
2335  =167F B0FF                    MOV     AL,true
2336  =1681 0AC0                    OR      AL,AL
2337  =1683 C3              ret4:   RET                             ;select disk function ok
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

```
2338
2339  =                     ;
2340  =                     home:
2341  =                             ;move to home position, then offset to start of dir
2342  =1684 E8910E    2518         CALL    homef
2343  =                                                     ;first directory pos. selected
2344  =1687 33C0                   XOR     AX,AX           ;constant zero to accumulator
2345  =1689 8B1EEE22               MOV     BX,curtrka
2346  =168D 8907                   MOV     [bx],ax                 ;curtrk=0000
2347  =                                                     ;curtrk, currec both set to 0
2348  =168F C3                     RET
2349  =                     ;
2350  =                     rdbuff:
2351  =                             ;read buffer and check if ok
2352  =1690 8B161A23               mov     dx,arecord
2353  =1694 E8900E    2527         CALL    readf           ;current drive, track, ...
2354  =1697 EB07      16A0         jmps    diocomp         ;check for i/o errors
2355  =                     ;
2356  =                     wrbufflg:
2357  =1699 8A0E0D23               mov     cl,wflag
2358  =                     wrbuff:
2359  =                             ;write buffer and check condition
2360  =                             ;write type (wrtype) is in register CL
2361  =                             ;wrtype = 0 => normal write operation
2362  =                             ;wrtype = 1 => directory write operation
2363  =                             ;wrtype = 2 => start of new block
2364  =169D E88A0E    252A         CALL    writef          ;current drive, track, ...
2365  =                     diocomp:                        ;check for disk errors
2366  =16A0 0AC0                   OR      AL,AL
2367  =16A2 74DF      1683         jz      ret4            ;rz
2368  =16A4 BB0422                 mov     BX,offset pererr
2369  =16A7 E96FFF    1619         JMP     goerr
2370  =                     ;
2371  =                     seekdir:
2372  =                             ;seek the record containing the current dir entry
2373  =16AA 8B1EA624               MOV     BX,dcnt         ;directory counter to BX
2374  =16AE B102                   MOV     CL,dskshf
2375  =16B0 D3EB                   shr     bx,cl           ;value to BX
2376  =16B2 891E1A23               MOV     arecord,BX
2377  =16B6 891E1F23               MOV     drec,BX         ;ready for seek
2378  =                                                     ; jmp seek
2379  =                                                     ;ret
2380  =                     ;
2381  =                     seek:
2382  =                             ;seek the track given by arecord (actual record)
2383  =                     ;
2384  =                     ;
2385  =16B6 A11A23                 mov     ax,arecord      ;compute track/sector
2386  =16B9 BA0000                 mov     dx,0
2387  =16BC F736FA22               div     sectpt          ;dx=sector, ax=track
2388  =16C0 52                     push    dx
2389  =16C1 8BC8                   mov     cx,ax           ;update curtrk
2390  =16C3 8B36EE22               mov     si,curtrka
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER #

```
2391
2392 =16CB 890C                    mov     [si],cx
2393 =16CD 030E0723                add     cx,offsetv              ;set BIOS track
2394 =16D1 E84A0E       251E       call    settrkf
2395 =16D4 59                      pop     cx                      ;set BIOS sector
2396 =16D5 8B160923                mov     dx,tranv
2397 =16D9 E8540E       2530       call    sectran
2398 =16DC 8BCB                    mov     cx,bx
2399 =16DE E9400E       2521       jmp     setsecf
2400 =                             ;ret
2401 =                        ;
2402 =                        ;
2403 =                        ;       utility functions for file access
2404 =                        ;
2405 =                        dmposition:
2406 =                               ;compute disk map position for vrecord to BX
2407 =16E1 BBFC22                  mov     bx,offset blkshf
2408 =16E4 8A0F                    MOV     CL,[bx]                 ;shift count to CL
2409 =16E6 A01823                  MOV     AL,lvrecord             ;current virtual record to A
2410 =16E9 D2E8                    shr     al,cl
2411 =                             ;A = shr(vrecord,blkshf) = vrecord/2**(sect/block)
2412 =16EB 8AE8                    MOV     CH,AL                   ;save it for later addition
2413 =16ED B107                    MOV     cl,7
2414 =16EF 2A0F                    SUB     cl,[bx]
2415 =16F1 A01723                  MOV     AL,extval               ;extent value ani extmsk
2416 =                        ;
2417 =                                                             ;blkshf = 3,4,5,6,7
2418 =                                                             ;CL=4,3,2,1,0
2419 =                                                             ;shift is 4,3,2,1,0
2420 =16F4 D2E0                    shl     al,cl
2421 =                        ;arrive here with A = shl(ext and extmsk,7-blkshf)
2422 =16F6 02C5                    ADD     AL,CH                   ;add the previous
2423 =                                                             ;shr(vrecord,blkshf) value
2424 =                                                             ;AL is one of the following
2425 =                                                             ;values, depending upon alloc
2426 =                                                             ;bks blkshf
2427 =                                                             ;1k   3     v/8 + extval * 16
2428 =                                                             ;2k   4     v/16+ extval * 8
2429 =                                                             ;4k   5     v/32+ extval * 4
2430 =                                                             ;8k   6     v/64+ extval * 2
2431 =                                                             ;16k  7     v/128+extval * 1
2432 =16F8 C3                      RET                             ;with dm$position in A
2433 =                        ;
2434 =                        getdm:
2435 =                               ;return disk map value from position given by CX
2436 =16F9 8B1E9924                MOV     BX,info                 ;base address of file
2437 =                                                             ;file control block
2438 =16FD 83C310                  ADD     BX,dskmap               ;bx=.diskmap
2439 =1700 03D9                    ADD     BX,CX                   ;index by asingle byte value
2440 =1702 803E122300              cmp     single,0                ;single byte/map entry?
2441 =1707 7405         170E       JZ      getdmd                  ;get disk map single byte
2442 =1709 8A1F                    MOV     BL,[bx]
2443 =170B B700                    MOV     BH,0
```

CP/M 86 V.1.1 - Vol I
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SCB # C81-162

```
2444
2445  =170D C3                          RET                             ;with BX=00bb
2446  =                           getdmd:
2447  =170E 03D9                        ADD     BX,CX                   ;bx=.fcb(dm+1*2)
2448  =                                                                 ;return double precision value
2449  =1710 8B1F                        MOV     bx,[bx]
2450  =1712 C3                          RET
2451  =                           ;
2452  =                           index:
2453  =                                 ;compute disk block number from current FCB
2454  =1713 E8C3FF      16E1           CALL    dmposition              ;0...15 in register AL
2455  =1716 8AC8                        MOV     CL,AL
2456  =1718 B500                        MOV     CH,0
2457  =171A E8DCFF      16F9           CALL    getdm                   ;value to BX
2458  =171D 891E1A23                    MOV     arecord,BX
2459  =1721 C3                          RET
2460  =                           ;
2461  =                           alloct:
2462  =                                 ;called following index to see if block allocated
2463  =1722 8B1E1A23                    MOV     BX,arecord
2464  =1726 0B0B                        or      bx,bx
2465  =1728 C3                          RET
2466  =                           ;
2467  =                           atran:
2468  =                                 ;compute actual record address, assuming index called
2469  =1729 8A0EFC22                    MOV     cl,blkshf               ;shift count to reg AL
2470  =172D 8B1E1A23                    MOV     BX,arecord
2471  =1731 D3E3                        shl     bx,cl
2472  =1733 891E1C23                    MOV     ablock,BX               ;save shifted block #
2473  =1737 A01823                      MOV     AL,lvrecord
2474  =173A 2206FD22                    AND     AL,blkmsk               ;masked value in AL
2475  =173E 0AD3                        OR      bl,al
2476  =1740 891E1A23                    MOV     arecord,BX              ;arecord=BX or
2477  =                                                                 ;(vrecord and blkmsk)
2478  =1744 C3                          RET
2479  =                           ;
2480  =                           getexta:
2481  =                                 ;get current extent field address to AL
2482  =1745 8B1E9924                    MOV     BX,info
2483  =1749 83C30C                      add     bx,extnum               ;bx=.fcb(extnum)
2484  =174C 8A07                        mov     al,[bx]
2485  =174E C3                          RET
2486  =                           ;
2487  =                           gtfcba:
2488  =                                 ;compute reccnt and nxtrec addresses for get/setfcb
2489  =174F BA0F00                      MOV     DX,reccnt
2490  =1752 03169924                    add     dx,info                 ;DX=.fcb(reccnt)
2491  =1756 BB1100                      MOV     BX,(nxtrec-reccnt)
2492  =1759 03DA                        ADD     BX,DX                   ;bx=.fcb(nxtrec)
2493  =175B C3                          RET
2494  =                           ;
2495  =                           getfcb:
2496  =                                 ;set variables from currently addressed FCB
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
2497
2498 =175C E8F0FF      174F        CALL    gtfcba              ;addresses in DX, BX
2499 =175F 8A07                  MOV     AL,[bx]
2500 =1761 A21823                MOV     lvrecord,al         ;vrecord=fcb(nxtrec)
2501 =1764 87DA                  XCHG    BX,DX
2502 =1766 8A07                  MOV     AL,[bx]
2503 =1768 A21623                MOV     rcount,AL           ;rcount=fcb(reccnt)
2504 =176B E8D7FF      1745      CALL    getexta             ;BX=.fcb(extnum)
2505 =176E A0FE22                MOV     AL,extmsk           ;extent mask to a
2506 =1771 2207                  AND     AL,[bx]             ;fcb(extnum) and extmsk
2507 =1773 A21723                MOV     extval,AL
2508 =1776 C3                    RET
2509 =                       ;
2510 =                       setfcb:
2511 =                           ;place values back into current fcb
2512 =1777 E8D5FF      174F      CALL    gtfcba              ;addresses to DX, BX
2513 =177A A00F23                MOV     AL,seqio
2514 =177D 3C02                  CMP     AL,02
2515 =177F 7502        1783      JNZ     setfc1
2516 =1781 32C0                  XOR     AL,AL               ;check ranfill
2517 =                       setfc1:
2518 =                                                       ;=1 if sequential i/o
2519 =1783 02061823              add     AL,lvrecord
2520 =1787 8807                  MOV     [bx],al             ;fcb(nxtrec)=vrecord+seqio
2521 =1789 87DA                  XCHG    BX,DX
2522 =178B A01623                MOV     AL,rcount
2523 =178E 8807                  MOV     [bx],al             ;fcb(reccnt)=rcount
2524 =1790 C3                    RET
2525 =                       ;
2526 =                       ;
2527 =                       cmpecs:
2528 =                           ;compute checksum for current directory buffer
2529 =1791 B98000                MOV     cx,recsiz           ;size of directory buffer
2530 =1794 8B1EF222              MOV     BX,buffa            ;current directory buffer
2531 =1798 32C0                  XOR     AL,AL               ;clear checksum value
2532 =                       cmpec0:
2533 =179A 0207                  ADD     AL,[bx]
2534 =179C 43                    INC     BX
2535 =179D E2FB        179A      loop    cmpec0
2536 =179F C3                    RET                         ;with checksum in A
2537 =                       ;
2538 =                       setcdisk:
2539 =                           ;set a '1' value in curdsk position of CX
2540 =17A0 51                    PUSH    CX                  ;save input parameter
2541 =17A1 8A0EA424              MOV     cl,curdsk
2542 =17A5 BB0100                MOV     BX,1                ;number to shift
2543 =17A8 D3E3                  shl     bx,cl               ;BX = mask to integrate
2544 =17AA 59                    POP     CX                  ;original mask
2545 =17AB 08D9                  or      bx,cx               ;BX = mask or rol(1,curdsk)
2546 =17AD C3                    RET
2547 =                       ;
2548 =                       nowrite:
2549 =                           ;return true if dir checksum difference occurred
```

COPYRIGHT (C) 1981, 1982
DIGITAL RESEARCH
PACIFIC GROVE, CA 93950

```
2550
2551 =17AE A1E822                    MOV     ax,rodsk
2552 =17B1 8A0EA424                  MOV     cl,curdsk
2553 =17B5 D3E8                      shr     ax,cl
2554 =17B7 2401                      AND     AL,1
2555 =17B9 C3                        RET                             ;non zero if nowrite
2556 =                          ;
2557 =                          setro:
2558 =                                                               ;set current disk to read only
2559 =17BA 8B0EE822                  mov     cx,rodsk
2560 =17BE E8D4FF       17A0         CALL    setcdisk                ;sets bit to 1
2561 =17C1 891EE822                  MOV     rodsk,BX
2562 =                                                               ;high water mark in directory
2563 =                                                               ;goes to max
2564 =17C5 8B160123                  MOV     dx,dirmax
2565 =17C9 42                        INC     dx
2566 =17CA 8B1EEC22                  MOV     BX,cdrmaxa              ;BX = .cdrmax
2567 =17CE 8917                      MOV     [bx],dx                 ;cdrmax = dirmax
2568 =17D0 C3                        RET
2569 =                          ;
2570 =                          ckrodir:
2571 =                                ;check current directory element for read/only status
2572 =17D1 E81B00       17EF         CALL    getdptra                ;address of element
2573 =                          ;
2574 =                          ckrofile:
2575 =                                ;check current buff(dptr) or fcb(0) for r/o status
2576 =17D4 BA0900                     MOV     DX,rofile
2577 =17D7 03DA                      ADD     BX,DX                   ;offset to r/o bit
2578 =17D9 8A07                      MOV     AL,[bx]
2579 =17DB D0D0                      RCL     AL,1
2580 =17DD 7318         17FA         jae     ret5                    ;rnc
2581 =17DF BB0A22                    mov     BX,offset rofer
2582 =17E2 EB08         17EC         jmps     jmp_goerr                   ;
2583 =                          ;       jmp roferror ;exit to read only disk message
2584 =                          ;
2585 =                          ;
2586 =                          checkwrite:
2587 =                                ;check for write protected disk
2588 =17E4 E8C7FF       17AE         CALL    nowrite
2589 =17E7 7411         17FA         jz      ret5                    ;rz
2590 =17E9 BB0822                    mov     BX,offset roderr
2591 =                          jmp_goerr:
2592 =17EC E92AFE       1619         JMP     goerr
2593 =                          ;       jmp roderror ;read only disk error
2594 =                          ;
2595 =                          getdptra:
2596 =                                ;compute the address of a directory element at
2597 =                                ;positon dptr in the buffer
2598 =17EF 8B1EF222                  MOV     BX,buffa
2599 =17F3 A01E23                    MOV     AL,dptr
2600 =                          addh:
2601 =                                ;BX = BX + AL
2602 =17F6 B400                      mov     ah,0
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

```
2603
2604 =17F8 03D8                    add     bx,ax
2605 =17FA C3              ret5:   ret
2606 =                     ;
2607 =                     ;
2608 =                     getmodnum:
2609 =                             ;compute the address of the module number
2610 =                             ;bring module number to accumulator
2611 =                             ;(high order bit is fwf (file write flag)
2612 =17FB 8B1E9924                MOV     BX,info
2613 =17FF 83C30E                  add     bx,modnum
2614 =1802 8A07                    MOV     AL,[bx]
2615 =1804 C3                      RET                     ;AL=fcb(modnum)
2616 =                     ;
2617 =                     clrmodnum:
2618 =                             ;clear the module number field for user open/make
2619 =1805 E8F3FF       17FB       CALL    getmodnum
2620 =1808 C60700                  MOV     b[bx],0         ;fcb(modnum)=0
2621 =180B C3                      RET
2622 =                     ;
2623 =                     setfwf:
2624 =180C E8ECFF       17FB       CALL    getmodnum       ;BX=.fcb(modnum),
2625 =                             ;AL=fcb(modnum)
2626 =                             ;set fwf(file write flag) to 1
2627 =180F 0C80                    OR      AL,fwfmsk
2628 =1811 8807                    MOV     [bx],al         ;fcb(modnum)=fcb(modnum) + 80h
2629 =                             ;also returns non zero
2630 =                             ;in accumulator
2631 =1813 C3                      RET
2632 =                     ;
2633 =                     ;
2634 =                     compcdr:
2635 =                             ;return cy if cdrmax > dcnt
2636 =1814 8B16A624                MOV     dx,dcnt         ;DX = directory counter
2637 =1818 3B1EEC22                MOV     BX,cdrmaxa      ;BX=.cdrmax
2638 =181C 3B17                    cmp     dx,[bx]
2639 =                             ;condition dcnt - cdrmax
2640 =                             ;produces cy if cdrmax>dcnt
2641 =181E C3              ret6:   RET
2642 =                     ;
2643 =                     setcdr:
2644 =                             ;if not (cdrmax > dcnt) then cdrmax = dcnt+1
2645 =181F E8F2FF       1814       CALL    compcdr
2646 =1822 72FA         181E       jb      ret6            ;return if cdrmax > dcnt
2647 =                             ;otherwise, BX = .cdrmax+1,
2648 =                             ;DX = dcnt
2649 =1824 42                      INC     DX
2650 =1825 8917                    MOV     [bx],dx
2651 =1827 C3                      RET
2652 =                     ;
2653 =                     subdh:
2654 =                             ;compute BX = DX - BX
2655 =1828 52                      push    dx
```

COPYRIGHT 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
2656
2657 =1829 2BD3                    sub     dx,bx
2658 =182B 8BDA                    mov     bx,dx
2659 =182D 5A                      pop     dx
2660 =182E C3                      RET
2661 =                          ;
2662 =                          newchecksum:
2663 =                                  ;drop through to compute new checksum
2664 =182F B1FF                        MOV     CL,true
2665 =                          checksum:
2666 =                                  ;compute current checksum record and update the
2667 =                                  ;directory element if CL=true, or check for = if not
2668 =                                  ;drec < chksiz?
2669 =1831 8B161F23                    MOV     DX,drec
2670 =1835 881E0523                    MOV     BX,chksiz
2671 =1839 E8ECFF       1828           CALL    subdh                   ;DX-BX
2672 =183C 73E0         181E           jae     ret6                    ;skip checksum if past
2673 =                                                                 ;checksum vector size
2674 =                                                                 ;drec < chksiz, so continue
2675 =183E 51                          PUSH    CX                      ;save init flag
2676 =183F E84FFF       1791           CALL    compcs                  ;check sum value to AL
2677 =1842 8B1E1F23                    MOV     BX,drec                 ;value of drec
2678 =1846 031EF822                    ADD     BX,checka               ;bx=.check(drec)
2679 =184A 59                          POP     CX                      ;recall true or false to CL
2680 =184B FEC1                        INC     CL                      ;true produces zero flag
2681 =184D 740D         185C           JZ      initcs
2682 =                                                                 ;not initializing, compare
2683 =184F 3A07                        CMP     AL,[bx]                 ;compute$cs=check(drec)?
2684 =1851 7403         185E           jz      ret7                    ;no message if ok
2685 =                                                                 ;checksum error, are we beyond
2686 =                                                                 ;the end of the disk?
2687 =1853 E8BEFF       1814           CALL    compcdr
2688 =1856 7306         185E           jae     ret7                    ;no message if so
2689 =1858 E85FFF       17BA           CALL    setro                   ;read/only disk set
2690 =185B C3                          RET
2691 =                          initcs:
2692 =                                                                 ;initializing the checksum
2693 =185C 8807                        MOV     [bx],al
2694 =185E C3                   ret7:  RET
2695 =                          ;
2696 =                          ;
2697 =                          wrdir:
2698 =                                  ;write the current directory entry, set checksum
2699 =185F E8CDFF       182F           CALL    newchecksum             ;initialize entry
2700 =1862 E81900       187E           CALL    setdir                  ;directory dma
2701 =1865 B101                        MOV     CL,1                    ;indicates a write directory
2702 =1867 E833FE       169D           CALL    wrbuff                  ;write the buffer
2703 =186A EB06         1872           jmps     setdata                 ;to data dma address
2704 =                                                                 ;ret
2705 =                          ;
2706 =                          rddirbuf:
2707 =                                  ;read a directory entry into the directory buffer
2708 =186C E80F00       187E           CALL    setdir                  ;directory dma
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
[illegible]
SER. #

```
2709
2710  =186F E81EFE      1690          CALL    rdbuff                  ;directory record loaded
2711  =                                                               ; jmp setdata
2712  =                                                               ; ret
2713  =                         ;
2714  =                         setdata:
2715  =                                 ;set data dma address
2716  =1872 8B0EA224                    mov     cx,dmabase
2717  =1876 E8BA0C      2533            call    setdmbf         ;set disk I/O base
2718  =1879 BBA024                      mov     BX,offset dmaad
2719  =187C EB08        1886            jmps    setdma                  ;to complete the call
2720  =                         ;
2721  =                         setdir:
2722  =                                 ;set directory dma address
2723  =187E 8CD9                        mov     cx,ds
2724  =1880 E8B00C      2533            call    setdmbf                 ;set BIOS disk I/O base
2725  =1883 BBF222                      mov     BX,offset buffa         ;jmp setdma to complete call
2726  =                         ;
2727  =                         setdma:
2728  =                                 ;BX=.dma address to set (i.e., buffa or dmaad)
2729  =1886 8B0F                        MOV     CX,[bx]                 ;parameter ready
2730  =1888 E9990C      2524            JMP     setdmf
2731  =                         ;
2732  =                         ;
2733  =                         dirtouser:
2734  =                                 ;copy the directory entry to the user buffer
2735  =                                 ;after call to search or searchn by user code
2736  =188B 8B16F222                    MOV     DX,buffa                ;source is directory buffer
2737  =188F 8B1EA024                    MOV     BX,dmaad                ;destination is user dma addr.
2738  =1893 B180                        MOV     CL,recsiz               ;copy entire record
2739  =1895 06                          push    es                      ;move to user segment
2740  =1896 8E069D24                    mov     es,parametersegment
2741  =189A E87EFD      161B            call    move
2742  =189D 07                          pop     es
2743  =189E CJ                          ret
2744  =                                                               ;ret
2745  =                         ;
2746  =                         endofdir:
2747  =                                 ;return zero flag if at end of director, non zero
2748  =                                 ;if not at end (end of dir if dcnt = 0ffffh)
2749  =189F BBA624                      mov     BX,offset dcnt
2750  =18A2 8A07                        MOV     AL,[bx]                 ;may be 0ffh
2751  =18A4 43                          INC     BX
2752  =18A5 3A07                        CMP     AL,[bx]                 ;low(dcnt) = high(dcnt)?
2753  =18A7 7502        18AB            jnz     ret8                    ;return non zero if different
2754  =                                                                 ;high and low the same,= 0ffh?
2755  =18A9 FEC0                        INC     AL                      ;0ffh becomes 00 if so
2756  =18AB C3                  ret8:   RET
2757  =                         ;
2758  =                         setenddir:
2759  =                                 ;set dcnt to the end of the directory
2760  =18AC BBFFFF                      MOV     BX,enddir
2761  =18AF 891EA624                    MOV     dcnt,bx
```

DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
2762
2763  =18B3 C3                          RET
2764  =                           ;
2765  =                           rddir:
2766  =                                 ;read next directory entry, with CL=true if initializing
2767  =18B4 8B160123                    MOV     DX,dirmax           ;in preparation for subtract
2768  =18B8 8B1EA624                    MOV     BX,dcnt
2769  =18BC 43                          INC     BX
2770  =18BD 891EA624                    MOV     dcnt,bx             ;dcnt=dcnt+1
2771  =                                                             ;continue while dirmax >= ( nt
2772  =                                                             ;(dirmax-dcnt no cy)
2773  =18C1 E864FF      1828            CALL    subdh               ;DX-BX
2774  =18C4 7302        18C8            JAE     rddir0
2775  =                                                             ;yes, set dcnt to end
2776  =                                                             ;of directory
2777  =18C6 EBE4        18AC            Jmps    setenddir
2778  =                           ;               ret
2779  =                           rddir0:
2780  =                                 ;not at end of directory, seek next element
2781  =                                                             ;CL=initialization flag
2782  =18C8 A0A624                      MOV     AL,ldcnt
2783  =18CB 2403                        AND     AL,dskmsk           ;low(dcnt) and dskmsk
2784  =18CD 51                          push    cx
2785  =18CE B105                        MOV     cl,fcbshf           ;to multiply by fcb size
2786  =18D0 D2E0                        shl     al,cl
2787  =18D2 59                          pop     cx
2788  =                                                             ;A = (low(dcnt) and dskmsk)
2789  =                                                             ;shl fcbshf
2790  =18D3 A21E23                      MOV     dptr,AL             ;ready for next dir operation
2791  =18D6 0AC0                        OR      AL,AL
2792  =18D8 7526        1900            jnz     ret71               ;return if not a new record
2793  =18DA 51                          PUSH    CX                  ;save initialization flag CL
2794  =18DB E8CCFD      16AA            CALL    seekdir             ;seek proper record
2795  =18DE E88BFF      186C            CALL    rddirbuf            ;read the directory record
2796  =18E1 59                          POP     CX                  ;recall initialization flag
2797  =18E2 E94CFF      1831            JMP     checksum            ;checksum the directory elt
2798  =                                                             ;ret
2799  =                           ;
2800  =                           ;
2801  =                                 if      not loader
2802  =                           getallocbit:
2803  =                                 ;given allocation vector
2804  =                                 ;position on CX, return byte
2805  =                                 ;containing CX shifted so that the least significant
2806  =                                 ;bit is in the low order accumulator position. BX is
2807  =                                 ;the address of the byte for
2808  =                                 ;possible replacement in
2809  =                                 ;memory upon return, and DH contains the number of shifts
2810  =                                 ;required to place the returned value back into position
2811  =                           ;
2812  =18E5 8BD1                        mov     dx,cx
2813  =18E7 80E107                      and     cl,111b
2814  =18EA FEC1                        INC     cl
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
[illegible]VE, CA 95030
SER. # =

```
2815
2816  =18EC 8AE9                mov     ch,cl
2817  =                                                 ;ch and cl both contain the
2818  =                                                 ;number of bit positions to
2819  =                                                 ;shift
2820  =18EE B103                mov     cl,3
2821  =18F0 D3EA                shr     dx,cl   ;shift bit address right 3 for byte address
2822  =                                                 ;dx shr 3 to dx
2823  =18F2 8B1EF822            MOV     BX,alloca       ;base addr. of alloc. vector
2824  =18F6 03DA                add     bx,dx
2825  =                                                 ;byte to A, hl =
2826  =18F8 8A07                mov     al,[bx]         ;.alloc(CX shr 3)
2827  =                                                 ;now move the bit to the
2828  =                                                 ;low order position of AL
2829  =18FA 8ACD                mov     cl,ch
2830  =18FC D2C0                rol     al,cl
2831  =18FE 8BD1                mov     dx,cx
2832  =                         endif
2833  =1900 C3          ret71:  ret
2834  =                         ;
2835  =                         if      not loader
2836  =                         ;
2837  =                         setallocbit:
2838  =                                 ;CX is the bit position of ALLOC to set or reset. The
2839  =                                 ;value of the bit is in register DL.
2840  =1901 52                  PUSH    DX
2841  =1902 E8E0FF      13E5    CALL    getallocbit     ;shifted val AL,count in DL
2842  =1905 24FE                AND     AL,11111110b    ;mask low bit to zero
2843  =                                                 ;(may be set)
2844  =1907 59                  POP     CX
2845  =1908 0AC1                OR      AL,CL           ;low bit of CL masked into AL
2846  =                                                 ; jmp rotr
2847  =                                                 ; ret
2848  =                         rotr:
2849  =                                 ;byte value from ALLOC is in register AL, with shift count
2850  =                                 ;in register CH (to place bit back into position), and
2851  =                                 ;target ALLOC position in registers BX, rotate and replace
2852  =                         ;
2853  =190A 51                  push    cx
2854  =190B 8ACE                mov     cl,dh
2855  =190D D2C8                ror     al,cl
2856  =190F 8807                mov     [bx],al
2857  =1911 59                  pop     cx
2858  =1912 C3                  ret
2859  =                         endif
2860  =                         ;
2861  =                         ;*************** end BDOS filesystem part 1 ***************
2862                            end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

```
2863
2864 =                    eject ! include bdosf2.a86     ; file system part 2
2865 =                    ;****************** BDOS file system part 2 ******************
2866 =                    ;
2867 =                    scndm:
2868 =                            ;scan the disk map addressed by dptr for non-zero
2869 =                            ;entries, the allocation vector entry corresponding
2870 =                            ;to a non-zero entry is set to the value of CL (0,1)
2871 =                            ;BX addresses the beginning of the directory entry
2872 =                    ;
2873 =1913 E8D9FE      17EF          CALL    getdptra                ;BX = buffa + dptr
2874 =1916 83C310                  ADD     BX,dskmap               ;BX = diskmap address
2875 =1919 51                      PUSH    CX                      ;save the 0/1 bit to set
2876 =191A B111                    MOV     CL,fcblen-dskmap+1      ;size of single byte diskmap+1
2877 =                     scndm0:
2878 =                                                              ;loop for each diskmap entry
2879 =191C 58                      POP     DX                      ;recall bit parity
2880 =191D FEC9                    DEC     CL
2881 =191F 7501          1922      JNZ     L_41
2882 =1921 C3                      RET
2883 =                     L_41:                                    ;all done scanning?
2884 =                                                              ;no, get next entry for scan
2885 =1922 52                      PUSH    DX                      ;replace bit parity
2886 =1923 803L122300              cmp     single,0
2887 =1928 7408          1932      JZ      scndm1
2888 =                                                              ;single byte scan operation
2889 =192A 51                      PUSH    CX                      ;save counter
2890 =192B 53                      PUSH    BX                      ;save map address
2891 =192C 8A0F                    MOV     CL,[bx]
2892 =192E B500                    MOV     CH,0                    ;CX=block#
2893 =1930 EB07          1939      jmps    scndm2
2894 =                     scndm1:
2895 =                                                              ;double byte scan operation
2896 =1932 FEC9                    DEC     CL                      ;count for double byte
2897 =1934 51                      PUSH    CX                      ;save counter
2898 =1935 8B0F                    MOV     cx,[bx]                 ;CX=block
2899 =1937 43                      INC     BX
2900 =1938 53                      PUSH    BX                      ;save map address
2901 =                     scndm2:
2902 =                                                              ;arrive here with CX=block#,
2903 =                                                              ;DL= 0/1
2904 =1939 0BC9                    OR      cx,cx                   ;skip if =0000
2905 =193B 7403          1943      JZ      scan3
2906 =193D 8B1EFF22                MOV     BX,maxall               ;check invalid index
2907 =1941 3BD9                    cmp     bx,cx
2908 =1943 7203          1948      JNAE    L_42
2909 =                             if      not loader
2910 =1945 E8B9FF        1901      CALL    setallocbit
2911 =                             endif
2912 =                     L_42:                                    ;
2913 =                                                              ;bit set to 0/1
2914 =                     scan3:                                   ;
2915 =1948 5B                      POP     BX
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
2916
2917  =1949 43                      INC     BX                      ;to next bit position
2918  =194A 59                      POP     CX                      ;recall counter
2919  =194B EBCF        191C        jmps    scndm0                  ;for another item
2920  =                       ;
2921  =                       initialize:
2922  =                               ;initialize the current disk
2923  =                               ;lret = false ;set to true if $ file exists
2924  =                               ;compute the length of the allocation vector - 2
2925  =194D 8B1EFF22                  MOV     BX,maxall
2926  =1951 B103                      MOV     CL,3                    ;perform maxall/8
2927  =1953 D3EB                      shr     bx,cl
2928  =1955 43                        INC     BX
2929  =1956 8BCB                      mov     cx,bx
2930  =1958 8B1EF822                  MOV     BX,alloca               ;base of allocation vector
2931  =                               ;fill the allocation vector
2932  =                               ;with zeros
2933  =                       init0:
2934  =195C C60700                    MOV     b[bx],0
2935  =195F 43                        INC     BX                      ;alloc(i)=0
2936  =1960 E2FA        195C          loop    init0
2937  =                               ;set the reserved space for
2938  =                               ;the directory
2939  =1962 8B160323                  MOV     DX,dirblk
2940  =1966 831EF822                  MOV     BX,alloca               ;BX=.alloc()
2941  =196A 8917                      MOV     [bx],dx                 ;sets reserved directory blks
2942  =                               ;allocation vector initialized
2943  =196C E815FD      1684          CALL    home                    ;home disk
2944  =                               ;cdrmax = 3 (scans at least
2945  =                               ;one directory record)
2946  =196F 8B1EEC22                  MOV     BX,cdrmaxa
2947  =1973 C7070300                  mov     w[bx],3
2948  =
2949  =1977 E832FF      18AC          CALL    setenddir               ;cdrmax = 0000
                                                                      ;dcnt = enddir
2950  =                               ;read directory entries and
2951  =                               ;check for allocated storage
2952  =                       init2:
2953  =197A B1FF                      MOV     CL,true
2954  =197C E835FF      18B4          CALL    rddir
2955  =197F E81DFF      189F          CALL    endofdir
2956  =1982 743B        19BF          jz      ret8b                   ;rz
2957  =                               ;not end of directory,
2958  =                               ;valid entry ?
2959  =1984 E868FE      17EF          CALL    getdptra                ;BX = buffa + dptr
2960  =1987 803FE5                    cmp     b[bx],empty
2961  =198A 74EE        197A          JZ      init2                   ;go get another item
2962  =                               ;not empty,user code the same?
2963  =198C A0A524                    MOV     AL,usrcode
2964  =198F 3A07                      CMP     AL,[bx]
2965  =1991 750C        199F          JNZ     pdollar
2966  =                               ;same user code,
2967  =                               ;check for '$' submit
2968  =1993 43                        INC     BX
```

SER. #

```
2969
2970 =1994 8A07                    MOV     AL,[bx]             ;first character
2971 =1996 2C24                    SUB     AL,'$'              ;dollar file?
2972 =1998 7505          199F      JNZ     pdollar
2973 =                                                         ;$ file found, mark in lret
2974 =199A FEC8                    DEC     AL
2975 =199C A29B24                  MOV     lret,al             ;lret = 255
2976 =                       pdollar:
2977 =                                                         ;now scan the disk map for
2978 =                                                         ;allocated blocks
2979 =199F B101                    MOV     CL,1                ;set to allocated
2980 =19A1 E86FFF        1913      CALL    scndm
2981 =19A4 E878FE        181F      CALL    setcdr              ;set cdrmax to dcnt
2982 =19A7 EBD1          197A      jmps    init2               ;for another entry
2983 =                       ;
2984 =                       copydirloc:
2985 =                             ;copy directory location to lret following
2986 =                             ;delete, rename, ... ops
2987 =19A9 A00E23                  MOV     AL,dirloc
2988 =19AC E945FC        15F4      JMP     staret
2989 =                       ;     ret
2990 =                       ;
2991 =                       compext:
2992 =                             ;compare extent# in AL with that in CL, return nonzero
2993 =                             ;if they do not match
2994 =19AF 51                      PUSH    CX                  ;save CL's original value
2995 =19B0 8A2EFE22                MOV     ch,extmsk
2996 =19B4 F6D5                    NOT     ch
2997 =                                                         ;CH has negated form of
2998 =                                                         ;extent mask
2999 =19B6 22CD                    and     cl,ch               ;low bits removed from CL
3000 =19B8 22C5                    AND     AL,CH               ;low bits removed from AL
3001 =19BA 2AC1                    SUB     AL,CL
3002 =19BC 241F                    AND     AL,maxext           ;set flags
3003 =19BE 59                      POP     CX                  ;restore original values
3004 =19BF C3                ret8b: RET
3005 =                       ;
3006 =                       ;
3007 =                       searchi:
3008 =                       ;     search initialization
3009 =19C0 8B1E9924                mov     bx,info
3010 =19C4 891E2123                mov     srcha,bx      ;searcha = info
3011 =                       searchi1:
3012 =19C8 C6060E23FF              mov     dirloc,0ffh
3013 =19CD 880EA824                mov     srchl,cl
3014 =19D1 C3                      ret
3015 =                       search:
3016 =                       srch:
3017 =                                                         ;search for directory element
3018 =                                                         ;of lenght
3019 =19D2 E8EBFF        19C0      call    searchi
3020 =19D5 E8D4FE        18AC      CALL    setenddir           ;dcnt = enddir
3021 =19D8 E8A9FC        1684      CALL    home                ;to start at the beginning
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
3022
3023  =                                                    ;(drop through to searchn)
3024  =                    ;
3025  =                    searchn:
3026  =                    srchn:
3027  =                            ;search for the next directory element, assuming
3028  =                            ;a previous call on search which sets searcha and searchhl
3029  =                            ;
3030  =19DB C6069A2300              mov     user_zero_pass,false
3031  =                    ;
3032  =19E0 B100                    MOV     CL,false
3033  =19E2 E8CFFE      18B4        CALL    rddir           ;read next dir element
3034  =19E5 E8B7FE      189F        CALL    endofdir
3035  =19E8 7503        19ED        jnz     srchn1
3036  =                    ;
3037  =                    jmp_srchfin:
3038  =19EA E98000      1A6D        jmp     srchfin
3039  =                    ;
3040  =                    srchn1:
3041  =                                                    ;skip to end if so
3042  =                                                    ;not end of directory,
3043  =                                                    ;scan for match
3044  =19ED 8B162123                MOV     dx,srcha        ;DX=beginning of user fcb
3045  =19F1 8BF2                    MOV     SI,DX
3046  =19F3 AC                      LODS    al              ;first character
3047  =19F4 3CE5                    CMP     AL,empty        ;keep scanning if empty
3048  =19F6 7407        19FF        JZ      srchnext
3049  =                                                    ;not empty, may be end of
3050  =                                                    ;logical directory
3051  =19F8 52                      PUSH    DX              ;save search address
3052  =19F9 E818FE      1814        CALL    compcdr         ;past logical end?
3053  =19FC 5A                      POP     DX              ;recall address
3054  =19FD 73EB        19EA        jnb     jmp_srchfin
3055  =                                                    ;artificial stop
3056  =                    srchnext:
3057  =19FF E8EDFD      17EF        CALL    getdptra        ;BX = buffa+dptr
3058  =1A02 8A0EA824                mov     cl,srchl        ;length of search to CL
3059  =1A06 B500                    MOV     CH,0            ;CH counts up, CL counts down
3060  =                    srchloop:
3061  =1A08 0AC9                    or      cl,cl
3062  =1A0A 7438        1A44        JZ      endsearch
3063  =1A0C 8BF2                    MOV     SI,DX
3064  =1A0E AC                      LODS    al
3065  =1A0F 3C3F                    CMP     AL,'?'
3066  =1A11 7429        1A3C        JZ      srchok          ;? matches all
3067  =                                                    ;scan next character if not
3068  =                                                    ;ubytes
3069  =1A13 80FD0B                  CMP     ch,ubytes
3070  =1A16 7424        1A3C        JZ      srchok
3071  =                                                    ;not the ubytes field,
3072  =                                                    ;extent field
3073  =1A18 80FD0C                  CMP     ch,extnum       ;may be extent field
3074  =1A1B 7408        1A25        JZ      srchext         ;skip to search extent
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
3075
3076 =1A1D 2A07                    SUB     AL,[bx]
3077 =1A1F 247F                    AND     AL,7fh              ;mask-out flags/extent modulus
3078 =1A21 7533          1A56      jnz     srchnm
3079 =1A23 EB17          1A3C      jmps    srchok              ;matched character
3080 =                   srchext:
3081 =                                                         ;A has fcb character
3082 =                                                         ;attempt an extent # match
3083 =1A25 51                      PUSH    CX                  ;save counters
3084 =1A26 8A0F                    MOV     C,[bx]              ;directory character to c
3085 =1A28 E884FF        17AF      CALL    compext             ;compare user/dir char
3086 =1A2B 59                      POP     CX                  ;recall counters
3087 =1A2C 9F                      lahf
3088 =1A2D F6069A23FF              test    user_zero_pass,true
3089 =1A32 7545          1A79      jnz     save_dcnt_pos2      ;save position o dcnt
3090 =                   ;
3091 =                   ;   disable search under user zero if any fcb found under this user
3092 =                   ;
3093 =1A34 C606972300              mov     search_user0,false
3094 =1A39 9E                      sahf
3095 =1A3A 759F          19DB      jnz     srchn               ;skip if no match
3096 =                   srchok:
3097 =                                                         ;current character matches
3098 =1A3C 42                      INC     DX
3099 =1A3D 43                      INC     BX
3100 =1A3E FEC5                    INC     CH
3101 =1A40 FEC9                    DEC     CL
3102 =1A42 EBC4          1A08      jmps    srchloop
3103 =                   endsearch:
3104 =                             ;entire name matches, return dir position
3105 =1A44 C6060E2300              mov     dirloc,0h           ;mark as found
3106 =1A49 8A0A24                  MOV     AL,ldcnt
3107 =1A4C 2403                    AND     AL,dskmsk           ;return code 0-3
3108 =1A4E A27B24                  MOV     lret,al
3109 =                                                         ;lret = low(dcnt) and 11b
3110 =1A51 8AE8                    mov     ch,al
3111 =1A53 FEC5                    inc     ch                  ;set the zero flag
3112 =1A55 C3            ret9:     RET
3113 =                   ;
3114 =                   srchnm:
3115 =1A56 0AED                    or      ch,ch          ;search 'NO' match routine
3116 =1A58 7530          1A8A      jnz     searchn_jmp
3117 =1A5A F607FF                  test    b[bx],true
3118 =1A5D 752B          1A8A      jnz     searchn_jmp
3119 =1A5F F6069723FF              test    search_user0,true
3120 =1A64 7424          1A8A      jz      searchn_jmp
3121 =1A66 C6069A23FF              mov     user_zero_pass,true
3122 =1A6B EBCF          1A3C      jmps    srchok
3123 =                   ;
3124 =                   srchfin:
3125 =                             ;end of directory, or empty name
3126 =1A6D E83CFE        18AC      CALL    setenddir           ;may be artifical end
3127 =1A70 B0FF                    MOV     AL,255
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
3128
3129  =1A72 8AEB                    mov     ch,al
3130  =1A74 FEC5                    inc     ch                      ;zero flag set on unsuccessful
3131  =1A76 E97BFB      15F4        JMP     staret
3132  =                         ;
3133  =                         ;
3134  =                         save_dcnt_pos2:
3135  =1A79 9E                      sahf
3136  =1A7A 750E        1A8A        jne     searchn_jmp
3137  =1A7C 43                      inc     bx
3138  =1A7D 803F00                  cmp     b[bx],0         ;check for modnum=0 under user 0
3139  =1A80 7508        1A8A        jne     searchn_jmp
3140  =1A82 8B36A624                mov     si,dcnt
3141  =1A86 89369823                mov     xdcnt,si
3142  =                         searchn_jmp:
3143  =1A8A E94EFF      19DB        jmp     searchn
3144  =                         ;
3145  =                             if      not loader
3146  =                         ;
3147  =                         delet:
3148  =                             ;delete the currently addressed file
3149  =1A8D E854FD      17E4        CALL    checkwrite              ;write protected?
3150  =1A90 B10C                    MOV     CL,extnum
3151  =1A92 E83DFF      19D2        CALL    srch                    ;search through file type
3152  =                         delet0:
3153  =                                                             ;loop while directory matches
3154  =1A95 E807FE      189F        CALL    endofdir
3155  =1A98 74BB        1A55        jz      ret9
3156  =                                                             ;set each non zero disk map
3157  =                                                             ;entry to 0
3158  =                                                             ;in the allocation vector
3159  =                                                             ;may be r/o file
3160  =1A9A E834FD      17D1        CALL    ckrodir                 ;ro disk error if found
3161  =1A9D E84FFD      17EF        CALL    getdptra                ;BX=.buff(dptr)
3162  =1AA0 C607E5                  MOV     b[bx],empty
3163  =1AA3 B100                    MOV     CL,0
3164  =1AA5 E868FE      1913        CALL    scndm                   ;alloc elts set to 0
3165  =1AA8 E8B4FD      185F        CALL    wrdir                   ;write the directory
3166  =1AAB E82DFF      19DB        CALL    srchn                   ;to next element
3167  =1AAE EBE5        1A95        jmps    delet0                  ;for another record
3168  =                             endif
3169  =                         ;
3170  =                             if      not loader
3171  =                         getblock:
3172  =                             ;given allocation vector position CX, find the zero bit
3173  =                             ;closest to this position by searching left and right.
3174  =                             ;if found, set the bit to 1 and return the bit position
3175  =                             ;in BX.  if not found (i.e., we pass 0 on the left, or
3176  =                             ;maxall on the right), return 0000 in BX
3177  =                         ;
3178  =1AB0 8BD1                    MOV     dx,cx                   ;copy start pos. to DX
3179  =                         lefttst:
3180  =1AB2 0BC9                    or      cx,cx
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
3181
3182 =1AB4 740C        1AC2        JZ      righttst        ;skip if left=0000
3183 =                                                      ;left not at position zero,
3184 =                                                      ;bit zero ?
3185 =1AB6 49                      DEC     CX
3186 =1AB7 52                      PUSH    DX
3187 =1AB8 51                      PUSH    CX              ;left,right pushed
3188 =1AB9 E829FE      18E5        CALL    getallocbit
3189 =1ABC D0D8                    RCR     AL,1
3190 =1ABE 7318        1AD8        JAE     rblok           ;return block number if zero
3191 =                                                      ;bit is one, so try the right
3192 =1AC0 59                      POP     CX
3193 =1AC1 5A                      POP     DX              ;left, right restored
3194 =                         righttst:
3195 =1AC2 3B16FF22                cmp     dx,maxall       ;value of maximum allocation#
3196 =1AC6 7318        1AE2        JAE     rblok0          ;return block 0000 if so
3197 =1AC8 42                      INC     DX
3198 =1AC9 51                      PUSH    CX
3199 =1ACA 52                      PUSH    DX              ;left, right pushed
3200 =1ACB 8BCA                    MOV     cx,dx           ;ready right for call
3201 =1ACD E815FE      18E5        CALL    getallocbit
3202 =1AD0 D0D8                    RCR     AL,1
3203 =1AD2 7304        1AD8        JAE     rblok           ;return block number if zero
3204 =1AD4 5A                      POP     DX
3205 =1AD5 59                      POP     CX              ;restore left and right ptr
3206 =1AD6 EBDA        1AB2        jmps    lefttst         ;for another attempt
3207 =                         rblok:
3208 =1AD8 D0D0                    RCL     AL,1
3209 =1ADA FEC0                    INC     AL              ;bit back into position
3210 =                                                     ;and set to 1
3211 =                                                     ;DL contains the number of
3212 =                                                     ;shifts required to reposition
3213 =1ADC E82BFE      190A        CALL    rotr            ;move bit back to position
3214 =                                                     ;and store
3215 =1ADF 5B                      POP     BX
3216 =1AE0 5A                      POP     DX              ;BX returned value,
3217 =                                                     ;DX discarded
3218 =1AE1 C3                      RET
3219 =                         rblok0:
3220 =                                                     ;cannot find an available
3221 =                                                     ;bit, return 0000
3222 =1AE2 0BC9                    or      cx,cx
3223 =1AE4 75CC        1AB2        jnz     lefttst
3224 =1AE6 BB0000                  MOV     BX,0000h
3225 =1AE9 C3                      RET
3226 =                             endif
3227 =                         ;
3228 =                         ;
3229 =                         ;
3230 =                         copyfcb:
3231 =                                                     ;copy the entire FCB
3232 =1AEA B100                    MOV     CL,0
3233 =1AEC B220                    MOV     DL,fcblen       ;start at 0, to fcblen-1
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
USER #

```
3234
3235 =                                                    ;       jmp copydir
3236 =                    ;
3237 =                    copydir:
3238 =                            ;copy fcb information starting
3239 =                            ;into the currently addressed directory entry
3240 =1AEE 52                     PUSH    DX              ;save length for later
3241 =1AEF B500                   MOV     CH,0            ;double index to CX
3242 =1AF1 8B169924               MOV     dx,info         ;BX = source for data
3243 =1AF5 03D1                   ADD     dx,CX           ;DX=.fcb(CL), source for copy
3244 =1AF7 E8F5FC       17EF      CALL    getdptra        ;BX=.buff(dptr), destination
3245 =1AFA 59                     POP     CX              ;DX=source, BX=dest, C=length
3246 =1AFB E81DFB       161B      CALL    move            ;data moved
3247 =                    seekcopy:
3248 =                            ;enter from close to seek and copy current element
3249 =1AFE E8A9FB       16AA      CALL    seekdir         ;to the directory  lement
3250 =1B01 E95B6D       105F      JMP     wrdir           ;write the directory element
3251 =                                                    ;ret
3252 =                    ;
3253 =                    ;
3254 =                            if      not loader
3255 =                    rename:
3256 =                            ;rename the file described by the first half of
3257 =                            ;the currently addressed file control block. the
3258 =                            ;new name is contained in the last half of the
3259 =                            ;currently addressed file conrol block. the file
3260 =                            ;name and type are changed, but the reel number
3261 =                            ;is ignored.  the user number is identical.
3262 =                    ;
3263 =1B04 E8DDFC       17E4      CALL    checkwrite      ;may be write protected
3264 =                                                    ;search up to the extent field
3265 =1B07 B10C                   MOV     CL,extnum
3266 =1B09 E8C6FE       19D2      CALL    srch
3267 =                                                    ;copy position 0
3268 =1B0C 8B1E9924               MOV     BX,info
3269 =1B10 8A07                   MOV     AL,[bx]         ;BX=.fcb(0), A=fcb(0)
3270 =1B12 83C310                 ADD     BX,dskmap       ;bx=.fcb(diskmap)
3271 =1B15 8807                   MOV     [bx],al         ;fcb(dskmap)=fcb(0)
3272 =                                                    ;assume the same disk
3273 =                                                    ;for new named file
3274 =                    renam0:
3275 =1B17 E885FD       189F      CALL    endofdir
3276 =1B1A 7501         1B1D      JNZ     L_51
3277 =1B1C C3                     ret10:  RET
3278 =                            L_51:                   ;stop at end of dir
3279 =                                                    ;not end of directory,
3280 =                                                    ;rename next element
3281 =1B1D E8B1FC       17D1      CALL    ckrodir         ;may be read-only file
3282 =1B20 B110                   MOV     CL,dskmap
3283 =1B22 B20C                   MOV     DL,extnum
3284 =1B24 E8C7FF       1AEE      CALL    copydir
3285 =                                                    ;element renamed, move to next
3286 =1B27 E8B1FE       19DB      CALL    srchn
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
[illegible]
[illegible] CA 93950
SER. #

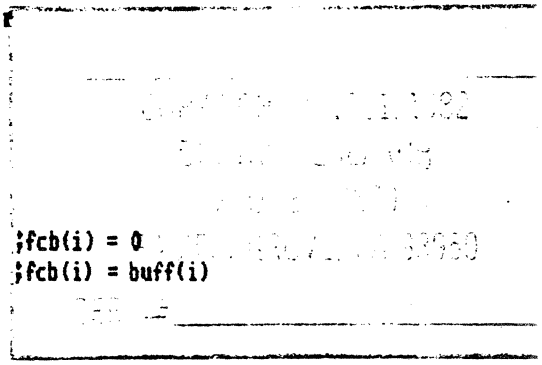
```
3287
3288 =1B2A EBEB        1B17        jmps    renam0
3289 =                           endif
3290 =                      ;
3291 =                           if      'loader
3292 =                      ret10:  ret
3293 =                           endif
3294 =                      ;
3295 =                      ;
3296 =                      ;
3297 =                      indicators:
3298 =                           ;set file indicators for current fcb
3299 =1B2C B10C                  MOV     CL,extnum
3300 =1B2E E8A1FE        19D2    CALL    srch            ;through file type
3301 =                      indic0:
3302 =1B31 E86BFD        189F    CALL    endofdir
3303 =1B34 74E6          1B1C    jz      ret10           ;stop at end of dir
3304 =                                                   ;not end of directory,
3305 =                                                   ;continue to change
3306 =1B36 B100                  MOV     CL,0
3307 =1B38 B20C                  MOV     DL,extnum       ;copy name
3308 =1B3A E8B1FF        1AEE    CALL    copydir
3309 =1B3D E893FE        19DB    CALL    srchn
3310 =1B40 EBEF          1B31    jmps    indic0
3311 =                      ;
3312 =                      open:
3313 =                           ;search for the directory entry, copy to FCB
3314 =1B42 B10F                  MOV     CL,namlen
3315 =1B44 E83BFE        19D2    CALL    srch
3316 =1B47 E855FD        189F    CALL    endofdir
3317 =                      open1:
3318 =1B4A 74D0          1B1C    jz      ret10           ;return with lret=255 if end
3319 =                                                   ;not end of directory,copy fcb
3320 =                      opencopy:
3321 =                                                   ;(referenced below to copy fcb
3322 =1B4C E8F6FB        1745    CALL    getexta
3323 =
3324 =1B4F 50                    PUSH    AX
3325 =1B50 53                    PUSH    BX              ;save extent#
3326 =1B51 E89BFC        17EF    CALL    getdptra
3327 =1B54 87DA                  XCHG    BX,DX           ;DX = .buff(dptr)
3328 =1B56 8B1E9924              MOV     BX,info         ;BX=.fcb(0)
3329 =1B5A B120                  MOV     CL,nxtrec       ;length of move operation
3330 =1B5C 52                    PUSH    DX              ;save .buff(dptr)
3331 =1B5D E8BBFA        161B    CALL    move            ;from .buff(dptr) to .fcb(0)
3332 =                                                   ;note that entire fcb is
3333 =                                                   ;copied, including indicators
3334 =1B60 E8A9FC        180C    CALL    setfwf          ;sets file write flag
3335 =1B63 5A                    POP     DX
3336 =1B64 8B0C00                MOV     BX,extnum
3337 =1B67 03DA                  ADD     BX,DX           ;bx=.buff(dptr+extnum)
3338 =1B69 8A0F                  mov     cl,[bx]
3339 =
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
3340
3341  =1B6B BB0F00              MOV     BX,reccnt
3342  =1B6E 03DA                ADD     BX,DX
3343  =1B70 8A2F                mov     ch,[bx]
3344  =
3345  =1B72 5B                  POP     BX
3346  =1B73 58                  POP     AX
3347  =1B74 8807                MOV     [bx],al             ;restore extent number
3348  =                 ;
3349  =                         ;BX = .user extent#, CH = dir rec cnt, CL = dir extent#
3350  =                         ;if user ext < dir ext then user := 128 records
3351  =                         ;if user ext = dir ext then user := dir records
3352  =                         ;if user ext > dir ext then user := 0 records
3353  =                 ;
3354  =1B76 3A0F                CMP     cl,[bx]
3355  =1B78 8AC5                MOV     A,CH                ;ready dir reccnt
3356  =1B7A 7406        1B82    JZ      openrcnt            ;if same, user gets dir reccnt
3357  =1B7C B000                MOV     AL,0
3358  =1B7E 7202        1B82    JB      openrcnt            ;user is larger
3359  =1B80 B080                MOV     AL,128              ;directory is larger
3360  =                 openrcnt:                           ;A has record count to fill
3361  =1B82 8B1E9924            MOV     BX,info
3362  =1B86 8A0F00              MOV     DX,reccnt
3363  =1B89 03DA                ADD     BX,DX
3364  =1B8B 8807                MOV     [bx],al
3365  =1B8D C3          ret11:  RET
3366  =                 ;
3367  =                 mergezero:
3368  =                                                     ;BX = .fcb1(i), DX = .fcb2(i),
3369  =                                                     ;if fcb1(i) = 0
3370  =                                                     ;then fcb1(i) := fcb2(i)
3371  =1B8E 833F00              cmp     w[bx],0
3372  =1B91 75FA        1B8D    jnz     ret11               ;nz
3373  =1B93 8BF2                MOV     SI,DX
3374  =1B95 AD                  LODS    ax
3375  =1B96 8907                mov     [bx],ax
3376  =1B98 C3          ret12:  RET
3377  =                 ;
3378  =                 close:
3379  =                         ;locate the directory element and re-write it
3380  =1B99 33C0                XOR     AX,AX
3381  =1B9B A29B24              MOV     lret,al
3382  =1B9E A3A624              MOV     dcnt,ax
3383  =1BA1 E80AFC      17AE    CALL    nowrite
3384  =1BA4 75F2        1B98    jnz     ret12               ;skip close if r/o disk
3385  =                                                     ;check file write flag -
3386  =                                                     ;0 indicates written
3387  =1BA6 E852FC      17FB    CALL    getmodnum           ;fcb(modnum) in A
3388  =1BA9 2480                AND     AL,fwfmsk
3389  =1BAB 75EB        1B98    jnz     ret12               ;return if bit remains set
3390  =1BAD B10F                MOV     CL,namlen
3391  =1BAF E820FE      19D2    CALL    srch                ;locate file
3392  =1BB2 E8EAFC      189F    CALL    endofdir
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
3393
3394  =1BB5 74E1        1B98        jz      ret12           ;return if not found
3395  =                             ;merge the disk map at info
3396  =1BB7 B91000                  MOV     CX,dskmap
3397  =1BBA E832FC      17EF        CALL    getdptra
3398  =1BBD 03D9                    ADD     BX,CX
3399  =1BBF 87DA                    XCHG    BX,DX           ;DX is .buff(dptr+16)
3400  =1BC1 8B1E9924                MOV     BX,info
3401  =1BC5 03D9                    ADD     BX,CX           ;dx=.buff(dptr+16),
3402  =                             ;bx=.fcb(16)
3403  =1BC7 B110                    MOV     CL,(fcblen-dskmap)  ;length of single byte dm
3404  =                         merge0:
3405  =1BC9 803E122300              cmp     single,0
3406  =1BCE 741B        1BEB        JZ      merged          ;skip to double
3407  =                         ;
3408  =                             ;this is a single byte map
3409  =                             ;if fcb(i) = 0 then fcb(i) = buff(i)
3410  =                             ;if buff(i) = 0 then buff(i) = fcb(i)
3411  =                             ;if fcb(i) <> buff(i) then error
3412  =                         ;
3413  =1BD0 8A07                    MOV     AL,[bx]
3414  =1BD2 0AC0                    OR      AL,AL
3415  =1BD4 8BF2                    MOV     SI,DX
3416  =1BD6 AC                      LODS    al
3417  =1BD7 7502        1BDB        JNZ     fcbnzero
3418  =                             ;fcb(i) = 0
3419  =1BD9 8807                    MOV     [bx],al         ;fcb(i) = buff(i)
3420  =                         fcbnzero:
3421  =1BDB 0AC0                    OR      AL,AL
3422  =1BDD 7506        1BE5        JNZ     buffnzero
3423  =                             ;buff(i) = 0
3424  =1BDF 8A07                    MOV     AL,[bx]
3425  =1BE1 8BFA                    MOV     DI,DX
3426  =1BE3 FC                      cld
3427  =1BE4 AA                      stos    al              ;buff(i)=fcb(i)
3428  =                         buffnzero:
3429  =1BE5 3A07                    CMP     AL,[bx]
3430  =1BE7 7547        1C30        jnz     mergerr         ;fcb(i) = buff(i)?
3431  =1BE9 EB16        1C01        jmps    dmset           ;if merge ok
3432  =                         merged:
3433  =                             ;this is a double byte merge
3434  =1BEB E8A0FF      1B8E        CALL    mergezero       ;buff = fcb if buff 0000
3435  =1BEE 87DA                    XCHG    BX,DX
3436  =1BF0 E89BFF      1B8E        CALL    mergezero
3437  =1BF3 87DA                    XCHG    BX,DX           ;fcb = buff if fcb 0000
3438  =                             ;they should be identical
3439  =                             ;at this point
3440  =1BF5 8BF2                    MOV     SI,DX
3441  =1BF7 8B04                    mov     ax,[si]
3442  =1BF9 3B07                    cmp     ax,[bx]
3443  =1BFB 7533        1C30        jnz     mergerr
3444  =1BFD 42                      inc     dx
3445  =1BFE 43                      inc     bx
```



```
3446
3447  =                                                    ;merge ok for this pair
3448  =1BFF FEC9                  DEC     CL               ;extra count for double byte
3449  =                     dmset:
3450  =1C01 42                    INC     DX
3451  =1C02 43                    INC     BX
3452  =1C03 FEC9                  DEC     CL
3453  =1C05 75C2        1BC9      JNZ     merge0           ;for more
3454  =                                                    ;end of disk map merge,
3455  =                                                    ;check record count
3456  =                                                    ;DX = .buff(dptr)+32,
3457  =                                                    ;BX = .fcb(32)
3458  =1C07 B9ECFF                MOV     CX,-(fcblen-extnum)
3459  =1C0A 03D9                  ADD     BX,CX
3460  =1C0C 87DA                  XCHG    BX,DX
3461  =1C0E 03D9                  ADD     BX,CX
3462  =                                                    ;DX = .fcb(extnum),
3463  =                                                    ;BX = .buff(dptr+extnum)
3464  =1C10 8BF2                  MOV     SI,DX
3465  =1C12 AC                    LODS    al               ;current user extent number
3466  =                                                    ;if fcb(ext) >= buff(.fcb) then
3467  =                                                    ;buff(ext) := fcb(ext),
3468  =                                                    ;buff(rec) := fcb(rec)
3469  =1C13 3A07                  CMP     AL,[bx]
3470  =1C15 7211        1C28      JB      endmerge
3471  =                                                    ;fcb extent number >=
3472  =                                                    ;dir extent number
3473  =1C17 8807                  MOV     [bx],al          ;buff(ext) = fcb(ext)
3474  =                                                    ;update dir. rec count field
3475  =1C19 B90300                MOV     CX,(reccnt-extnum)
3476  =1C1C 03D9                  ADD     BX,CX
3477  =1C1E 87DA                  XCHG    BX,DX
3478  =1C20 03D9                  ADD     BX,CX
3479  =
3480  =                                                    ;DX=.buff(reccnt),
3481  =1C22 8A07                  MOV     AL,[bx]          ;BX=.fcb(reccnt)
3482  =1C24 8BFA                  MOV     DI,DX
3483  =1C26 FC                    cld
3484  =1C27 AA                    stos    al               ;buff(reccnt)=fcb(reccnt)
3485  =                     endmerge:
3486  =1C28 C6060823FF            MOV     fcbcopied,true   ;mark as copied
3487  =1C2D E9CEFE        1AFE    JMP     seekcopy         ;ok to "wrdir" here -
3488  =                                                    ;1.4 compatible
3489  =                                                    ; ret
3490  =                     mergerr:
3491  =                                                    ;elements did not merge ok
3492  =1C30 F6E9B24               dec     lret             ;=255 non zero flag set
3493  =1C34 C3              ret13:  RET
3494  =                     ;
3495  =                     ;
3496  =                             if      not loader
3497  =                     make:
3498  =                             ;create a new file by creating a directory entry
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
3499
3500 =                                                  ;then opening the file
3501 =1C35 E8ACFB      17E4    CALL    checkwrite      ;may be write protected
3502 =1C38 FF369924            push    info
3503 =1C3C C7069924E722        MOV     info,offset efcb ;info = .empty
3504 =1C42 B101                MOV     CL,1
3505 =1C44 E888FD      19D2    CALL    srch            ;length 1 match on empty entry
3506 =1C47 E855FC      189F    CALL    endofdir        ;zero flag set if no space
3507 =1C4A 8F069924            pop     info            ;recall word ptr info address
3508 =1C4E 74E4        1C34    jz      ret13           ;return with error condition
3509 =                                                  ;255 if not found
3510 =1C50 8B169924            mov     dx,info
3511 =                                                  ;clear the remainder of fcb
3512 =1C54 BB0F00              MOV     BX,namlen
3513 =1C57 03DA                ADD     BX,DX           ;bx=.fcb(namlen)
3514 =1C59 B111                MOV     CL,fcblen-namlen ;number of bytes to fill
3515 =1C5B 32C0                XOR     AL,AL           ;clear AL for fill
3516 =                    make0:
3517 =1C5D 8807                MOV     [bx],al
3518 =1C5F 43                  INC     BX
3519 =1C60 FEC9                DEC     CL
3520 =1C62 75F9        1C5D    JNZ     make0
3521 =1C64 B80D00              MOV     BX,ubytes
3522 =1C67 03DA                ADD     BX,DX   ;bx=.fcb(ubytes)
3523 =1C69 8807                MOV     [bx],al         ;fcb(ubytes) = 0
3524 =1C6B E8B1FB      181F    CALL    setcdr          ;may have extended dir
3525 =                                                  ;now copy entry to dir
3526 =1C6E E879FE      1AEA    CALL    copyfcb
3527 =                                                  ;set file write flag to '1'
3528 =1C71 E998FB      180C    JMP     setfwf
3529 =                                                  ;ret
3530 =                         endif
3531 =                    ;
3532 =                    ;
3533 =                    openreel:
3534 =                         ;close the current extent, and open the next one
3535 =                         ;if possible.  RMF is true if in read mode.
3536 =1C74 32C0                XOR     AL,AL
3537 =1C76 A20B23              MOV     fcbcopied,AL    ;set true if actually copied
3538 =1C79 E81DFF      1B99    CALL    close           ;close current extent
3539 =                                                  ;lret remains at enddir
3540 =                                                  ;if we cannot open next ext
3541 =1C7C E820FC      189F    CALL    endofdir
3542 =1C7F 74B3        1C34    jz      ret13           ;return if end
3543 =                                                  ;increment extent number
3544 =1C81 8B1E9924            MOV     BX,info
3545 =1C85 83C30C              ADD     BX,extnum       ;BX=.fcb(extnum)
3546 =1C88 8A07                MOV     AL,[bx]
3547 =1C8A FEC0                INC     AL
3548 =1C8C 241F                AND     AL,maxext
3549 =1C8E 8807                MOV     [bx],al         ;fcb(extnum)=++1
3550 =1C90 740F        1CA1    JZ      openmod         ;move to next module if 0
3551 =                                                  ;may be in same extent group
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
3552
3553 =1C92 8AE8                    MOV     CH,AL
3554 =1C94 A0FE22                  MOV     AL,extmsk
3555 =1C97 22C5                    AND     AL,CH
3556 =                                                      ;if 0, then not in same group
3557 =1C99 22060B23                and     al,fcbcopied    ;true if fcb was copied to dir
3558 =                                                      ;puts 0 in AL if not written
3559 =1C9D 740D        1CAC        JZ      openr0          ;go to next physical extent
3560 =                                                      ;result is non zero,
3561 =                                                      ;so we must be in same extent
3562 =1C9F EB26        1CC7        jmps    openr1          ;to copy fcb info
3563 =                         openmod:
3564 =                                 ;extent number overflow, go to next module
3565 =1CA1 83C302                  ADD     BX,(modnum-extnum)      ;bx=.fcb(modnum)
3566 =1CA4 FE07                    INC     b[bx]           ;fcb(modnum)=++1
3567 =                                                      ;module number incremented,
3568 =                                                      ;check for overflow
3569 =1CA6 8A07                    MOV     AL,[bx]
3570 =1CA8 240F                    AND     AL,maxmod       ;mask high order bits
3571 =1CAA 7426        1CD2        JZ      openerr         ;cannot overflow to 0
3572 =                                                      ;otherwise, ok to continue
3573 =                                                      ;with new module
3574 =                         openr0:
3575 =1CAC B10F                    MOV     CL,namlen
3576 =1CAE E821FD      19D2        CALL    srch            ;next extent found?
3577 =1CB1 E8EBFB      189F        CALL    endofdir
3578 =1CB4 7511        1CC7        JNZ     openr1
3579 =                                                      ;end of file encountered
3580 =1CB6 A00C23                  MOV     AL,rmf
3581 =1CB9 FEC0                    INC     AL              ;0ffh becomes 00 if read
3582 =1CBB 7415        1CD2        JZ      openerr         ;sets lret = 1
3583 =                                                      ;try to extend current file
3584 =                             if      not loader
3585 =1CBD E875FF      1C35        CALL    make
3586 =                                                      ;cannot be end of directory
3587 =1CC0 E8DCFB      189F        CALL    endofdir
3588 =1CC3 740D        1CD2        JZ      openerr         ;with lret = 1
3589 =1CC5 EB03        1CCA        jmps    openr2
3590 =                             endif
3591 =                         ;
3592 =                         ;
3593 =                         openr1:
3594 =
3595 =1CC7 E882FE      1B4C        CALL    opencopy        ;not end of file, open
3596 =                         openr2:
3597 =1CCA E8FFFA      175C        CALL    getfcb          ;set parameters
3598 =1CCD 32C0                    XOR     AL,AL
3599 =1CCF E922F9      15F4        JMP     staret          ;lret = 0
3600 =                                                      ; ret
3601 =                         openerr:
3602 =                                                      ;cannot move to next extent
3603 =                                                      ;of this file
3604 =1CD2 E823F9      15F8        CALL    setlret1        ;lret = 1
```

CP/M 86 V.1.1 - Vol I
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER # C81-162

```
3605
3606  =1CD5 E934FB      180C        JMP     setfwf              ;ensure that it will not
3607  =                                                         ;be closed
3608  =                                                         ;ret
3609  =                     ;
3610  =                     ;***************** end BDOS file system part 2 ****************
3611                        end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
3612
3613 =                    eject ! include bdosf3.a86     ; file system part 3
3614 =                    ;********************* BDOS filesystem part 3 ******************
3615 =                    ;
3616 =                    ;
3617 =                    ;
3618 =                    sqread:
3619 =                           ;sequential disk read operation
3620 =1CD8 C6060F2301              MOV     seqio,1
3621 =                                                          ;drop through to diskread
3622 =                    ;
3623 =                    diskread:                             ;(may enter from seqdiskread)
3624 =1CDD C6060C23FF              MOV     rmf,true              ;read mode flag =
3625 =                                                          ;true (open$reel)
3626 =                                                          ;read the next record from
3627 =                                                          ;the current FCB
3628 =1CE2 E877FA       175C       CALL    getfcb                ;sets parameters for the read
3629 =1CE5 A01823                 MOV     AL,lvrecord
3630 =1CE8 3A061623               CMP     AL,rcount             ;vrecord-rcount
3631 =                                                          ;skip if rcount > vrecord
3632 =1CEC 7218         1D06      JB      recordok
3633 =                                                          ;not enough records in extent
3634 =                                                          ;record count must be 128
3635 =                                                          ;to continue
3636 =1CEE 3C80                   CMP     AL,128                ;vrecord = 128?
3637 =1CF0 7528         1D1A      JNZ     diskeof               ;skip if vrecord<>128
3638 =1CF2 E87FFF       1C74      CALL    openreel              ;go to next extent if so
3639 =1CF5 E857FA       174F      call    gtfcba                ;get address of next record
3640 =1CF8 33C0                   xor     ax,ax
3641 =1CFA A31823                 MOV     vrecord,ax             ;vrecord=00
3642 =1CFD 8807                   mov     [bx],al               ;zero record number
3643 =                                                          ;now check for open ok
3644 =1CFF 803E9B2400             cmp     lret,0                ;stop at eof
3645 =1D04 7514         1D1A      JNZ     diskeof
3646 =                    recordok:
3647 =                           ;arrive with fcb addressing a record to read
3648 =1D06 E80AFA       1713      CALL    index
3649 =                                                          ;error if reading
3650 =                                                          ;unwritten data
3651 =                                                          ;(returns 1 to be compatible
3652 =                                                          ;with 1.4)
3653 =1D09 E816FA       1722      CALL    alloct                ;arecord=0000?
3654 =1D0C 740C         1D1A      JZ      diskeof
3655 =                                                          ;record has been allocated,
3656 =                                                          ;read it
3657 =1D0E E818FA       1729      CALL    atran                 ;arecord now a disk address
3658 =1D11 E8A6F9       16BA      CALL    seek                  ;to proper track,sector
3659 =1D14 E879F9       1690      CALL    rdbuff                ;to dma address
3660 =1D17 E95DFA       1777      JMP     setfcb                ;replace parameter
3661 =                    ;       ret
3662 =                    diskeof:
3663 =1D1A E9DBF8       15F8      JMP     setlret1              ;lret = 1
3664 =                                                          ;ret
```

DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
3665
3666 =                      ;
3667 =                              if      not loader
3668 =                      sqwrite:
3669 =                              ;sequential disk write
3670 =1D1D C6060F2301               MOV     seqio,1
3671 =                                                      ;drop through to diskwrite
3672 =                      ;
3673 =                      diskwite:                       ;(may enter here from
3674 =                                                      ;seqdiskwrite above)
3675 =1D22 C6060C2300               MOV     rmf,false       ;read mode flag
3676 =                                                      ;write record to currently
3677 =                                                      ;selected file
3678 =1D27 E8BAFA       17E4        CALL    checkwrite      ;in case write protected
3679 =1D2A 8B1E9924                 MOV     BX,info         ;BX = .fcb(0)
3680 =1D2E E8A3FA       17D4        CALL    ckrofile        ;may be a read-only file
3681 =1D31 E828FA       175C        CALL    getfcb          ;to set local parameters
3682 =1D34 A01823                   MOV     AL,lvrecord
3683 =1D37 3C80                     CMP     AL,lstrec+1     ;vrecord-128
3684 =                                                      ;skip if vrecord > lstrec
3685 =                                                      ;vrecord = 128, cannot open
3686 =                                                      ;next extent
3687 =1D39 7203         1D3E        JB      dskwr0
3688 =1D3B E9BAF8       15F8        JMP     setlret1
3689 =                      dskwr0:                         ;lret=1
3690 =                                                      ;continue to write next record
3691 =1D3E E8D2F9       1713        CALL    index
3692 =1D41 E8DEF9       1722        CALL    alloct
3693 =1D44 B100                     MOV     CL,0            ;marked as normal write
3694 =                                                      ;operation for wrbuff
3695 =1D46 7548         1D90        jnz     dskwr1
3696 =                              ;not allocated:
3697 =                              ;the argument to getblock is the starting
3698 =                              ;position for the disk search, and should be
3699 =                              ;the last allocated block for this file, or
3700 =                              ;the value 0 if no space has been allocated
3701 =                      ;
3702 =1D48 E896F7       16E1        CALL    dmposition
3703 =1D4B A21123                   MOV     dminx,AL        ;save for later
3704 =1D4E 33C9                     xor     cx,cx           ;may use block zero
3705 =1D50 0AC0                     OR      AL,AL
3706 =1D52 7408         1D5C        JZ      nopblock        ;skip if no previous block
3707 =                                                      ;previous block exists at A
3708 =1D54 8AC3                     MOV     CL,AL
3709 =1D56 49                       dec     CX
3710 =1D57 E897FF9      16F9        CALL    getdm           ;previous block # to BX
3711 =1D5A 8BCB                     MOV     CX,BX           ;CX=prev block#
3712 =                      nopblock:
3713 =                                                      ;CX = 0, or previous block #
3714 =1D5C E851FD       1AB0        CALL    getblock        ;block # to BX
3715 =                                                      ;arrive with block# or zero
3716 =1D5F 0BDB                     or      bx,bx
3717 =1D61 7505         1D68        JNZ     blockok
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
3718
3719  =                                                     ;cannot find block to allocate
3720  =1D63 B002                    MOV     AL,2
3721  =1D65 E98CF8        15F4      JMP     staret          ;lret=2
3722  =                         blockok:
3723  =                                                     ;BX contains allocated block #
3724  =1D68 891E1A23                MOV     arecord,BX
3725  =1D6C 87DA                    XCHG    BX,DX           ;block number to DX
3726  =1D6E 8B1E9924                MOV     BX,info
3727  =1D72 83C310                  ADD     BX,dskmap       ;bx=.fcb(diskmap)
3728  =1D75 803E122300              cmp     single,0        ;set flags for single byte dm
3729  =1D7A A01123                  MOV     AL,dminx        ;recall dm index
3730  =1D7D B400                    mov     ah,0
3731  =1D7F 7406          1D87      JZ      allocwd         ;skip if allocating word
3732  =                                                     ;allocating a byte value
3733  =1D81 03D8                    add     bx,ax
3734  =1D83 8817                    MOV     [bx],dl         ;single byte alloc
3735  =1D85 EB07          1D8E      jmps    dskwru          ;to continue
3736  =                         allocwd:
3737  =                                                     ;allocate a word value
3738  =1D87 03D8                    ADD     BX,ax
3739  =1D89 03D8                    ADD     BX,ax           ;bx=.fcb(dminx*2)
3740  =1D8B 8917                    MOV     [bx],dx         ;double wd
3741  =1D8D 43                      INC     BX
3742  =                         dskwru:
3743  =                             ;disk write to previously unallocated block
3744  =1D8E B102                    MOV     CL,2            ;marked as unallocated write
3745  =                         dskwr1:
3746  =                             ;continue the write operation of no allocation error
3747  =                             ;C = 0 if normal write, 2 if to prev unalloc block
3748  =                         ;
3749  =1D90 803E7B2400              cmp     lret,0
3750  =1D95 7401          1D98      JZ      dskwr10
3751  =1D97 C3                      RET
3752  =                         dskwr10:
3753  =1D98 880E0D23                mov     wflag,cl        ;save the write flag
3754  =1D9C E88AF9        1729      CALL    atran           ;arecord set
3755  =1D9F 803E0F2302              cmp     seqio,2         ;check random with 0 fill
3756  =1DA4 7545          1DEB      jnz     dskw11
3757  =                         ;
3758  =                         ;             this is random write with zero fill
3759  =                         ;
3760  =1DA6 803L0D2302              cmp     wflag,2         ;check for new allocation
3761  =1DAB 753E          1DEB      JNZ     dskw11          ;old allocation
3762  =                         ;
3763  =                         ;       new alloction for random with zero fill
3764  =                         ;
3765  =1DAD FF361A23                push    arecord
3766  =1DB1 883EF222                MOV     di,buffa
3767  =1DB5 33C0                    xor     ax,ax           ;zero and fill
3768  =1DB7 B94000                  mov     cx,64           ;zero 64 words in dirbuf
3769  =1DBA FC                      cld
3770  =1DBB F3AB            rep     stos    ax
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
3771
3772  =1DBD E8BEFA      187E        CALL    setdir
3773  =1DC0 8B1E1C23               MOV     BX,ablock
3774  =1DC4 891E1A23       fill1:  MOV     arecord,BX
3775  =1DC8 E8EFF8      16BA        CALL    seek
3776  =                                                ;reg CL used for deblocking
3777  =1DCB E8CBF8      1699        CALL    wrbufflg   ;write fill record
3778  =1DCE 8B1E1A23               MOV     BX,arecord ;restore last record
3779  =1DD2 C6060D2300             mov     wflag,0      ;change  allocate flag
3780  =1DD7 8AC3                   mov     al,bl      ;arecord in BX
3781  =1DD9 8A36FD22               mov     dh,blkmsk
3782  =1DDD 22C6                   and     al,dh
3783  =1DDF 43                     inc     bx
3784  =1DE0 3AC6                   cmp     al,dh
3785  =1DE2 75E0        1DC4       JNZ     fill1      ;cont until cluster is zeroed
3786  =1DE4 8F061A23               pop     arecord
3787  =1DE8 E887FA      1872       CALL    setdata
3788  =                    dskw11:
3789  =1DEB E8CCF8      16BA       CALL    seek       ;to proper file position
3790  =                                                ;(CL=2 if new block,0 if not)
3791  =1DEE E8A8F8      1699       CALL    wrbufflg   ;written to disk
3792  =                                                ;increment record count
3793  =                                                ;if rcount <= vrecord
3794  =1DF1 A01823               MOV     AL,lvrecord
3795  =1DF4 BB1623               mov     BX,offset rcount
3796  =1DF7 3A07                 CMP     AL,[bx]    ;vrecord-rcount
3797  =1DF9 7206        1E01     JB      dskwr2
3798  =                                                ;rcount <= vrecord
3799  =1DFB 8807                 MOV     [bx],al
3800  =1DFD FE07                 INC     b[bx]      ;rcount = vrecord+1
3801  =1DFF B102                 MOV     CL,2       ;mark as record count
3802  =                                                ;incremented
3803  =                    dskwr2:
3804  =                                                ;AL has vrecord,
3805  =                                                ;CL=2 if new block or record#
3806  =1E01 50                   PUSH    AX         ;save vrecord value
3807  =1E02 E8F6F9      17FB     CALL    getmodnum  ;BX=.fcb(modnum),
3808  =                                                ;AL=fcb(modnum)
3809  =                                                ;reset the file write flag to
3810  =                                                ;mark as written FCB
3811  =1E05 247F                 AND     AL,not fwfmsk ;bit reset
3812  =1E07 8807                 MOV     [bx],al    ;fcb(modnum) = fcb(modnum)
3813  =                                                ;and 7fh
3814  =1E09 58                   POP     AX         ;restore vrecord
3815  =                    noupdate:
3816  =                          ;check for end of extent, if found attempt to open
3817  =                          ;next extent in preparation for next write
3818  =1E0A 3C7F                 CMP     AL,lstrec  ;vrecord=lstrec?
3819  =1E0C 751E        1E2C     JNZ     dskwr3     ;skip if not
3820  =                                                ;may be random access write,
3821  =                                                ;if so we are done
3822  =                                                ;change next
3823  =1E0E 803E0F2301           cmp     seqio,1
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

```
3824
3825 =1E13 7517        1E2C        JNZ     dskwr3              ;skip next extent open op
3826 =                                                       ;update current fcb before
3827 =                                                       ;going to next extent.
3828 =1E15 E85FF9      1777        CALL    setfcb
3829 =1E18 E859FE      1C74        CALL    openreel            ;rmf=false
3830 =                                                       ;vrecord remains at lstrec
3831 =                                                       ;causing eof if no more dir
3832 =                                                       ;space is available
3833 =1E1B BB9B24                  mov     BX,offset lret
3834 =1E1E 8A07                    MOV     AL,[bx]
3835 =1E20 0AC0                    OR      AL,AL
3836 =1E22 7505        1E29        JNZ     nospace
3837 =                                                       ;space available,
3838 =                                                       ;set vrecord = 255
3839 =1E24 FEC8                    dec     AL
3840 =1E26 A21823                  MOV     lvrecord,al         ;goes to 00 next time
3841 =                        nospace:
3842 =1E29 C60700                  MOV     b[bx],0             ;lret = 00 for returned value
3843 =                        dskwr3:
3844 =1E2C E948F9      1777        JMP     setfcb              ;replace parameters
3845 =                                                       ;ret
3846 =                        ;
3847 =                        rseek:
3848 =                              ;random access seek operation, C=0ffh if read mode
3849 =                              ;fcb is assumed to address an active FCB
3850 =                              ;(modnum has been set to 1100$0000b if previous bad seek)
3851 =                        ;
3852 =1E2F C6060F2300              MOV     seqio,0             ;marked as random access op.
3853 =                        rseek1:
3854 =1E34 51                      PUSH    CX                  ;save r/w flag
3855 =1E35 8B169924                MOV     DX,info
3856 =                                                       ;dx will hold base of fcb
3857 =1E39 B82100                  MOV     BX,ranrec
3858 =1E3C 03DA                    ADD     BX,DX               ;bx=.fcb(ranrec)
3859 =1E3E 8A07                    MOV     AL,[bx]
3860 =1E40 247F                    AND     AL,7fh
3861 =1E42 50                      PUSH    AX                  ;record number
3862 =1E43 8A07                    MOV     AL,[bx]
3863 =1E45 D0D0                    RCL     AL,1                ;cy=lsb of extent#
3864 =1E47 43                      INC     BX
3865 =1E48 8A07                    MOV     AL,[bx]
3866 =1E4A D0D0                    RCL     AL,1
3867 =1E4C 241F                    AND     AL,11111b           ;AL=ext#
3868 =1E4E 8AC8                    MOV     CL,AL               ;CL holds extent number
3869 =                                                       ;record stacked
3870 =1E50 8A07                    MOV     AL,[bx]
3871 =1E52 D0D8                    RCR     AL,1
3872 =1E54 D0D8                    RCR     AL,1
3873 =1E56 D0D8                    RCR     AL,1
3874 =1E58 D0D8                    RCR     AL,1
3875 =1E5A 240F                    AND     AL,1111b            ;mod#
3876 =1E5C 8AE8                    MOV     CH,AL               ;CH = module#,CL = ext#
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
3877
3878  =1E5E 58                      POP     AX                ;recall sought record #
3879  =                                                       ;check to insure high byte
3880  =1E5F 43                      INC     BX
3881  =1E60 8A1F                    MOV     BL,[bx]           ;l=high byte (must be 00)
3882  =1E62 0ADB                    or      bl,bl
3883  =1E64 B306                    MOV     BL,6              ;zero flag, l=6
3884  =                                                       ;produce error 6,
3885  =                                                       ;seek past physical eod
3886  =1E66 7569        1ED1        JNZ     seekerr
3887  =                                                       ;otherwise, high byte = 0
3888  =1E68 8B2000                  MOV     BX,nxtrec
3889  =1E6B 03DA                    ADD     BX,DX             ;bx=.fcb(nxtrec)
3890  =1E6D 8807                    MOV     [bx],al           ;sought rec# stored away
3891  =                     ;
3892  =                             ;a rive here with CH=mod#, CL=ext#, DX=.fcb, rec stored
3893  =                             ;the r/w flag is still stacked.  compare fcb values
3894  =                     ;
3895  =1E6F 8B0C00                  MOV     BX,extnum
3896  =1E72 03DA                    ADD     BX,DX
3897  =1E74 8AC1                    MOV     AL,CL             ;A=seek ext#
3898  =1E76 2A07                    SUB     AL,[bx]
3899  =1E78 750D        1E87        JNZ     ranclose          ;tests for = extents
3900  =                                                       ;extents match, check mod#
3901  =1E7A 8B0E00                  MOV     BX,modnum
3902  =1E7D 03DA                    ADD     BX,DX
3903  =1E7F 8AC5                    MOV     AL,CH             ;B=seek mod#
3904  =                                                       ;could be overflow at eof,
3905  =                                                       ;producing module# of 90h
3906  =                                                       ;or 10h, so compare all
3907  =                                                       ;but fwf
3908  =1E81 2A07                    SUB     AL,[bx]
3909  =1E83 247F                    AND     AL,7fh
3910  =1E85 743C        1EC3        JZ      seekok            ;same?
3911  =                     ranclose:
3912  =1E87 51                      PUSH    CX
3913  =1E88 52                      PUSH    DX                ;save seek mod#,ext#, .fcb
3914  =1E89 E80DFD      1B99        CALL    close             ;current extent closed
3915  =1E8C 5A                      POP     DX
3916  =1E8D 59                      POP     CX                ;recall parameters and fill
3917  =1E8E B303                    MOV     BL,3              ;cannot close error #3
3918  =1E90 A09B24                  MOV     AL,lret
3919  =1E93 FEC0                    INC     AL
3920  =1E95 7432        1EC9        JZ      badseek
3921  =1E97 8B0C00                  MOV     BX,extnum
3922  =1E9A 03DA                    ADD     BX,DX
3923  =1E9C 880F                    MOV     [bx],cl           ;fcb(extnum)=ext#
3924  =1E9E BB0L00                  MOV     BX,modnum
3925  =1EA1 03DA                    ADD     BX,DX
3926  =1EA3 882F                    MOV     [bx],ch           ;fcb(modnum)=mod#
3927  =1EA5 E89AFC      1B42        CALL    open              ;is the file present?
3928  =1EA8 A09B24                  MOV     AL,lret
3929  =1EAB FEC0                    INC     AL
```

COPYRIGHT (C) 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
3930
3931  =1EAD 7514        1EC3          JNZ     seekok                  ;open successful?
3932  =                                                               ;cannot open file, read mode?
3933  =1EAF 59                        POP     CX                      ;r/w flag to c (=0ffh if read)
3934  =1EB0 51                        PUSH    CX                      ;everyone expects this
3935  =                                                               ;item stacked
3936  =1EB1 B304                      MOV     BL,4                    ;seek to unwritten extent #4
3937  =1EB3 FEC1                      INC     CL                      ;becomes 00 if read operation
3938  =1EB5 7412        1EC9          JZ      badseek                 ;error if read operation
3939  =                                                               ;write, make new extent
3940  =1EB7 E878FD      1C35          CALL    make
3941  =1EBA B305                      MOV     BL,5                    ;cannot create new extent #5
3942  =1EBC A07B24                    MOV     AL,lret
3943  =1EBF FEC0                      INC     AL
3944  =1EC1 7406        1EC9          JZ      badseek         ;no dir space
3945  =                                                               ;file make successful
3946  =                         seekok:
3947  =1EC3 59                        POP     CX                      ;discard r/w flag
3948  =1EC4 32C0                      XOR     AL,AL
3949  =1EC6 E92BF7      15F4          JMP     staret                  ;with zero set
3950  =                         badseek:
3951  =                         ;
3952  =                               ;fcb no longer contains a valid fcb, mark
3953  =                               ;with 1100$000b in modnum field so that it
3954  =                               ;appears as overflow with file write flag set
3955  =                         ;
3956  =1EC9 53                        PUSH    BX                      ;save error flag
3957  =1ECA E82EF9      17FB          CALL    getmodnum               ;BX = .modnum
3958  =1ECD C607C0                    MOV     b[bx],11000000b
3959  =1ED0 5B                        POP     BX                      ;and drop through
3960  =                         seekerr:
3961  =1ED1 59                        POP     CX                      ;discard r/w flag
3962  =1ED2 8AC3                      MOV     AL,BL
3963  =1ED4 A27B24                    MOV     lret,al                 ;lret=#, nonzero
3964  =                                                               ;setfwf returns <> 0 if error
3965  =1ED7 E932F9      180C          JMP     setfwf                  ;flag set, so subsequent
3966  =                                                               ;close ok
3967  =                                                               ;ret
3968  =                         ;
3969  =                         rdread:
3970  =                         ;
3971  =                               ;random disk read operation
3972  =                         ;
3973  =1EDA B1FF                      MOV     CL,true                 ;marked as read operation
3974  =1EDC E850FF      1E2F          CALL    rseek
3975  =1EDF 7503        1EE4          JNZ     L_65
3976  =1EE1 E8F9FD      1CDD          CALL    diskread
3977  =                         L_65:                                 ;if seek successful
3978  =1EE4 C3                        RET
3979  =                         ;
3980  =                         rdwrite:
3981  =                                                               ;random disk write operation
3982  =1EE5 B100                      MOV     CL,false                ;marked as write operation
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
3983
3984  =1EE7 E845FF      1E2F        CALL    rseek
3985  =1EEA 7503        1EEF        JNZ     L_66
3986  =1EEC E833FE      1D22        CALL    diskwite
3987  =                       L_66:                           ;if seek successful
3988  =1EEF C3                    RET
3989  =                       ;
3990  =                       cmperr:
3991  =                               ;compute random record position for getfilesize/setrandom
3992  =1EF0 87DA                      XCHG    BX,DX
3993  =1EF2 03DA                      ADD     BX,DX
3994  =                                                       ;DX=.buf(dptr) or .fcb(0),
3995  =                                                       ;BX=.f(nxtrec/reccnt)
3996  =1EF4 8A0F                      MOV     CL,[bx]         ;pick up record no.
3997  =1EF6 B500                      MOV     CH,0            ;CX = 0000 0000 ?rrr rrrr
3998  =1EF8 B80C00                    MOV     BX,extnum
3999  =1EFB 03DA                      ADD     BX,DX
4000  =1EFD 8A27                      MOV     ah,[bx]
4001  =1EFF B000                      mov     al,0
4002  =1F01 D1E8                      shr     ax,1
4003  =1F03 80E40F                    and     ah,0fh
4004  =1F06 03C3                      add     cx,ax
4005  =
4006  =1F08 B80E00                    MOV     BX,modnum
4007  =1F0B 03DA                      ADD     BX,DX
4008  =1F0D 8A07                      MOV     AL,[bx]         ;A=XXX? mmmm
4009  =1F0F B410                      mov     ah,16
4010  =1F11 F6E4                      mul     ah
4011  =1F13 50                      PUSH    AX
4012  =1F14 02E8                      ADD     ch,al
4013  =                                                       ;cy=?,CX = mmmm eeee errr rrrr
4014  =1F16 9C                      pushf                   ;possible second carry
4015  =1F17 5B                      POP     BX              ;cy = lsb of L
4016  =1F18 8AC3                    MOV     AL,BL           ;cy = lsb of A
4017  =1F1A 5B                      POP     BX              ;cy = lsb of L
4018  =1F1B 0AC3                    OR      AL,BL           ;cy/cy = lsb of A
4019  =1F1D 2401                    AND     AL,1            ;Al=0? possible carry-out
4020  =1F1F C3                      RET
4021  =                       ;
4022  =                       getfilesize:
4023  =                               ;compute logical file size for current fcb
4024  =1F20 B10C                      MOV     CL,extnum
4025  =1F22 E8ADFA      19D2        CALL    srch
4026  =                                                       ;zero receiving ranrec field
4027  =1F25 8B1E9924                  MOV     BX,info
4028  =1F29 BA2100                    MOV     DX,ranrec
4029  =1F2C 03DA                      ADD     BX,DX
4030  =1F2E 53                      PUSH    BX              ;save position
4031  =1F2F 8837                    MOV     [bx],dh
4032  =1F31 43                      INC     BX
4033  =1F32 8837                    MOV     [bx],dh
4034  =1F34 43                      INC     BX
4035  =1F35 8837                    MOV     [bx],dh         ;=00 00 00
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER #

```
4036
4037  =                     getsize:
4038  =1F37 E865F9    189F        CALL    endofdir
4039  =1F3A 7422      1F5E        JZ      setsize
4040  =                                                   ;current fcb addressed by dptr
4041  =1F3C E8B0F8    17EF        CALL    getdptra
4042  =1F3F BA0F00                MOV     DX,reccnt       ;ready for compute size
4043  =1F42 E8ABFF    1EF0        CALL    cmperr
4044  =                                                   ;AL=0000 000?,
4045  =                                                   ;CX = mmmm eeee errr rrrr
4046  =                                                   ;compare with memory, larger?
4047  =1F45 5B                    POP     BX
4048  =1F46 53                    PUSH    BX              ;recall, replace .fcb(ranrec)
4049  =1F47 8AD0                  MOV     DL,AL           ;save cy
4050  =1F49 8BC1                  mov     ax,cx
4051  =1F4B 2B07                  sub     ax,[bx]
4052  =1F4D 8AC2                  MOV     AL,DL
4053  =1F4F 1A4702                SBB     AL,2[bx]        ;carry if .fcb(ranrec) >
4054  =                                                   ;directory
4055  =1F52 7205      1F59        JB      getnextsize     ;for another try
4056  =                                                   ;fcb is less or equal,
4057  =                                                   ;fill from directory
4058  =1F54 885702                MOV     2[bx],dl
4059  =1F57 890F                  MOV     [bx],cx
4060  =                     getnextsize:
4061  =1F59 E87FFA    19DB        CALL    srchn
4062  =1F5C EBD9      1F37        jmps    getsize
4063  =                     setsize:
4064  =1F5E 5B                    POP     BX              ;discard .fcb(ranrec)
4065  =1F5F C3                    RET
4066  =                     ;
4067  =                     setrandom:
4068  =                             ;set random record from the current FCB
4069  =1F60 8B1E9924              MOV     BX,info
4070  =1F64 BA2000                MOV     DX,nxtrec       ;ready params for computesize
4071  =1F67 E886FF    1EF0        CALL    cmperr          ;DX=info, AL=cy,
4072  =                                                   ;CX=mmmm eeee errr rrrr
4073  =1F6A BB2100                MOV     BX,ranrec
4074  =1F6D 03DA                  ADD     BX,DX           ;bx=.fcb(ranrec)
4075  =1F6F 890F                  MOV     [bx],cx
4076  =1F71 884702                MOV     2[bx],al        ;to ranrec
4077  =1F74 C3                    RET
4078  =                     ;
4079  =                             endif
4080  =                     ;
4081  =                     ;
4082  =                     ;
4083  =                     ;
4084  =                     ;
4085  =                     slect:
4086  =                             ;select disk word ptr info for subsequent input or output ops
4087  =1F75 8B1EEA22              MOV     BX,dlog
4088  =1F79 8A0EA424              MOV     cl,curdsk
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4089
4090  =1F7D D3CB                    ror     bx,cl
4091  =1F7F 53                      PUSH    BX
4092  =1F80 87DA                    XCHG    BX,DX               ;save it for test below,
4093  =                                                         ;send to seldsk
4094  =1F82 E8A2F6      1627        CALL    selectdisk
4095  =1F85 5B                      POP     BX                  ;recall dlog vector
4096  =1F86 7503        1F8B        JNZ     L_67
4097  =1F88 E88BF6      1616        CALL    selerror
4098  =                         L_67:                           ;returns true if select ok
4099  =                                                         ;is the disk logged in?
4100  =1F8B D0DB                    RCR     bl,1
4101  =1F8D 7267        1FF6        jb      ret20               ;rc
4102  =                                                         ;return if bit is set
4103  =                                                         ;disk not logged in,
4104  =                                                         ;set bit and initialize
4105  =1F8F 8B0EEA22                mov     cx,dlog
4106  =1F93 E80AF8      17A0        CALL    setcdisk
4107  =1F96 891EEA22                MOV     dlog,BX             ;dlog=set$cdisk(dlog)
4108  =1F9A E9B0F9      194D        JMP     initialize
4109  =                                                         ;ret
4110  =                         ;
4111  =                         curselect:
4112  =1F9D A01023                  MOV     AL,linfo
4113  =1FA0 A2A424                  mov     curdsk,al           ;curdsk=info
4114  =1FA3 EBD0        1F75        jmps    slect               ;ret
4115  =
4116  =                         parsave33r:
4117  =1FA5 E85CEC      0C04        call    parsave33
4118  =1FA8 EB03        1FAD        jmps    reselect
4119  =                         ;
4120  =                         parsave36r:
4121  =1FAA E85CEC      0C09        CALL    parsave36
4122  =                         ;
4123  =                         ;
4124  =                         reselect:
4125  =                                 ;check current fcb to see if reselection necessary
4126  =1FAD 8B1E9924                mov     bx,info             ;examine attribute bits
4127  =1FB1 C606962300              mov     high_ext,0
4128  =1FB6 F6470880                test    f8[bx],interfc_bit          ;fcb(8) and 80h
4129  =
4130  =1FBA 7409        1FC5        jz      reselect0
4131  =1FBC 80670B7F                and     f8[bx],not_interfc_bit      ;change FCB copy
4132  =1FC0 C606962360              mov     high_ext,zero_readro        ;then high_ext = 60h
4133  =                         reselect0:
4134  =1FC5 C6061323FF              MOV     resel,true          ;mark possible reselect
4135  =1FCA 8B1E9924                MOV     BX,info
4136  =1FCE 8A07                    MOV     AL,[bx]             ;drive select code
4137  =1FD0 241F                    AND     AL,00011111b        ;non zero is auto drive select
4138  =1FD2 FEC8                    dec     AL                  ;drive code normalized to
4139  =                                                         ;0...30, or 255
4140  =1FD4 A21023                  MOV     linfo,AL            ;save drive code
4141  =1FD7 3C1E                    CMP     AL,30
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4142
4143 =1FD9 7312          1FED          JAE     noselect
4144 =                                                      ;auto select function,
4145 =                                                      ;save curdsk
4146 =1FDB A0A424                     MOV     AL,curdsk
4147 =1FDE A21423                     MOV     olddsk,AL     ;olddsk=curdsk
4148 =1FE1 8A07                       MOV     AL,[bx]
4149 =1FE3 A21523                     MOV     fcbdsk,AL     ;save drive code
4150 =1FE6 24E0                       AND     AL,11100000b
4151 =1FE8 8807                       MOV     [bx],al       ;preserve hi bits
4152 =1FEA E8B0FF          1F9D       CALL    curselect
4153 =                           noselect:
4154 =                                                      ;set user code
4155 =1FED A0A524                     MOV     AL,usrcode    ;0...31
4156 =1FF0 8B1E9924                   MOV     BX,info
4157 =1FF4 0807                       OR      b[bx],al
4158 =1FF6 C3                    ret20: RET
4159 =                           ;
4160 =                           ;
4161 =                           check_fcb:
4162 =1FF7 803E962360                 cmp     high_ext,zero_readro  ;was file opened under user 0
4163 =1FFC 7507            2005       jne     xcheck_fcb
4164 =1FFE 8B1E9924                   mov     bx,info
4165 =2002 C60700                     mov     b[bx],0h      ;zero the user number
4166 =                           xcheck_fcb:
4167 =2005 C3                         ret
4168 =                           ;
4169 =                           ;
4170 =                           ;
4171 =                           ;
4172 =                           ;       individual function handlers
4173 =                           ;func12:
4174 =                           ;
4175 =                                   ;return version number
4176 =                           ;
4177 =                           ;
4178 =                           func13:
4179 =                           ;
4180 =                                   ;reset disk system - initialize to disk 0
4181 =                           ;
4182 =2006 F7060222FFFF               test    bioscs,0ffffh
4183 =200C 7504            2012       jnz     func13a
4184 =200E 8C0E0222                   mov     bioscs,cs
4185 =                           func13a:
4186 =2012 33DB                       xor     bx,bx
4187 =2014 891EE822                   MOV     rodsk,BX
4188 =2018 891EEA22                   MOV     dlog,BX
4189 =201C 8A1EB724                   mov     bl,curdrvs    ;allow sys load from sys drive
4190 =2020 881EA424                   MOV     curdsk,bl     ;note that usrcode remains
4191 =                                                      ;unchanged
4192 =2024 E94EFF          1F75       JMP     slect
4193 =                                                      ;ret ;jmp goback
4194 =                           ;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4195
4196  =  1F9D                  func14  EQU     curselect               ;
4197  =                        ;
4198  =                                ;select disk info
4199  =                        ;
4200  =                        ;
4201  =                        func15:
4202  =                        ;
4203  =                                ;open file
4204  =                        ;
4205  =2027 E87BFF      1FA5          CALL    parsave33r              ;copy FCB from user seg,
4206  =202A E8D8F7      1805          CALL    clrmodnum               ;clear the module number
4207  =202D 803EA52400                cmp     usrcode,0h              ;is current user # 0
4208  =2032 740B        203F          je      call_open
4209  =2034 C7069923FE00              mov     found_in_zero,not_found ;flag to indicate file found
4210  =203A C606972FFF                mov     search_user0,true       ;flat to indicate search user 0
4211  =                        ;
4212  =                        call_open:
4213  =203F E800FB      1B42          call    open
4214  =2042 E83C00      2081          call    openx                   ;returns iif unsuccessful
4215  =2045 F6069723FF                test    search_user0,true       ;do we search user 0?
4216  =                                                                ;don't search if any FCB
4217  =                                                                ;on other user
4218  =204A 7501        204D          jnz     try_zero
4219  =                        ret30:
4220  =204C C3                        ret                             ;yes, open failed
4221  =                        try_zero:
4222  =204D C606972300                mov     search_user0,false      ;reset switch
4223  =2052 813E9923FE00              cmp     found_in_zero,not_found ;file exists under user 0?
4224  =2058 74F2        204C          je      ret30
4225  =
4226  =205A 8B1E9823                  mov     bx,xdcnt                ;reset dcnt
4227  =205E 80E3FC                    and     bl,0fch                 ;set dcnt to last entry of
4228  =                                                                ;previous record
4229  =2061 4B                        dec     bx                      ;force read of dir record if change
4230  =2062 891EA624                  mov     dcnt,bx
4231  =2066 C606962360                mov     high_ext,zero_readro    ;set mode to read/only
4232  =                        open_user_zero:
4233  =206B 8B1E9924                  mov     bx,info
4234  =206F C60700                    mov     b[bx],0h                ;mov zero user # to fcb
4235  =2072 B10F                      mov     cl,namlen
4236  =2074 E849F9      19C0          call    searchi
4237  =2077 E861F9      19DB          call    searchn
4238  =207A E8CDFA      1B4A          call    open1
4239  =207D E80100      2081          call    openx
4240  =2080 C3                        ret                             ;unsuccessful under user 0
4241  =                        ;
4242  =                        ;
4243  =                        openx:
4244  =2081 E81BF8      189F          call    end_of_dir              ;open successful??
4245  =2084 74C6        204C          jz      ret30
4246  =2086 5B                        pop     bx                      ;discard return address
4247  =2087 803E962360                cmp     high_ext,60h            ;is open mode read/only???
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

```
4248
4249  =                                                    ;and user 0
4250  =208C 7512        20A0        jne     openx0
4251  =208E 8B1E9924                mov     bx,info
4252  =2092 8A470A                  mov     al,10[bx]      ;byte t2 in FCB
4253  =2095 2480                    and     al,80h         ;check for system file
4254  =2097 7507        20A0        jnz     openx0
4255  =2099 A29623                  mov     high_ext,al    ;open failed...not system
4256  =209C B0FF                    mov     al,0ffh
4257  =209E EB7E        211E        jmps    to_staret
4258  =                       openx0:
4259  =20A0 C3                      ret                    ;return from open function
4260  =                       ;
4261  =                                                    ;ret ;jmp goback
4262  =                       ;
4263  =                       func16:
4264  =                       ;
4265  =                             ;close file
4266  =                       ;
4267  =20A1 E801FF      1FA5        CALL    parsave33r
4268  =20A4 E7F2FA      1B99        jmp     close
4269  =                                                    ;ret ;jmp goback
4270  =                       ;
4271  =                       func17:
4272  =                             ;search for first occurrence of a file
4273  =                       ;
4274  =20A7 E85AEB      0C04        call    parsave33
4275  =                       ;
4276  =20AA 8B1E9D24                mov     bx,parametersegment    ;save DS seg containing srcha
4277  =20AE 871EAB24                mov     srcha_seg,bx
4278  =                       ;
4279  =20B2 B100                    MOV     CL,0           ;length assuming '?' true
4280  =20B4 8B1E9924                mov     bx,info
4281  =20B8 803F3F                  cmp     b[bx],'?'      ;no reselect if ?
4282  =20BB 7410        20CD        JZ      qselect        ;skip reselect if so
4283  =                                                    ;normal search
4284  =20BD E835F6      1745        CALL    getexta
4285  =20C0 803F3F                  cmp     b[bx],'?'      ;check extent for '?'
4286  =20C3 7403        20C8        jz      all_modules
4287  =20C5 E83DF7      1805        CALL    clrmodnum      ;module number zeroed
4288  =                       all_modules:
4289  =20C8 E8E2FE      1FAD        CALL    reselect
4290  =20CB B10F                    MOV     CL,namlen
4291  =                       qselect:
4292  =20CD E802F9      19D2        CALL    srch
4293  =20D0 EB16        20E8        jmps     to_dirtouser          ;copy directory entry to user
4294  =                                                    ;ret ;jmp goback
4295  =                       ;
4296  =                       func18:
4297  =                       ;
4298  =                             ;search for next occurrence of a file name
4299  =                             if      loader
4300  =                             ret
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER #_____

```
4301
4302  =                         endif
4303  =                    ;
4304  =                         if      not loader
4305  =                    ;
4306  =20D2 8B1EAB24            mov     bx,srcha_seg          ;user's last search segment
4307  =20D6 891E7D24            mov     parametersegment,bx
4308  =20DA 8B1EA924            MOV     BX,info_save          ;user's last search address
4309  =20DE 891E9924            MOV     info,bx
4310  =20E2 E8C0FE      1FA5    call    parsave33r
4311  =20E5 E8F3F8      19DB    call    srchn                 ;search for next
4312  =                         endif
4313  =                    to_dirtouser:
4314  =20E8 E9A0F7      188B    jmp     dirtouser             ;copy directory entry to user
4315  =                    ;
4316  =                                                      ;ret ;jmp goback
4317  =                    ;
4318  =                    func19:
4319  =                    ;
4320  =                         ;delete a file
4321  =                         if      loader
4322  =                         ret
4323  =                         endif
4324  =                    ;
4325  =                         if      not loader
4326  =                    ;
4327  =20EB E8B7FE      1FA5    CALL    parsave33r
4328  =20EE E89CF9      1A8D    CALL    delet
4329  =20F1 E9B5F8      19A9    jmp     cpydirloc
4330  =                                                      ;ret ;jmp goback
4331  =                         endif
4332  =                    ;
4333  =                    ;
4334  =                    func20:
4335  =                    ;
4336  =                         ;read a file
4337  =                    ;
4338  =20F4 E8ACFE      1FA5    CALL    parsave33r
4339  =20F7 E8FDFE      1FF7    call    check_fcb
4340  =20FA E9DBFB      1CD8    jmp     sqread
4341  =                                                      ;jmp goback
4342  =                    ;
4343  =                    func21:
4344  =                         if      loader
4345  =                         ret
4346  =                         endif
4347  =                    ;
4348  =                    ;
4349  =                         if      not loader
4350  =                    ;
4351  =                         ;write a file
4352  =                    ;
4353  =20FD E8A5FE      1FA5    CALL    parsave33r
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4354
4355  =2100 E91AFC      1D1D        jmp     sqwrite
4356  =                                                     ;jmp goback
4357  =                             endif
4358  =                       ;
4359  =                       ;
4360  =                       func22:
4361  =                       ;
4362  =                             ;make a file
4363  =                             if      loader
4364  =                             ret
4365  =                             endif
4366  =                       ;
4367  =                             if      not loader
4368  =2103 E89FFE      1FA5        CALL    parsave33r
4369  =2106 E8FCF6      1805        CALL    clrmodnum
4370  =2109 E929FB      1C35        jmp     make
4371  =                                                     ;ret ;jmp goback
4372  =                             endif
4373  =                       ;
4374  =                       func23:
4375  =                       ;
4376  =                             ;rename a file
4377  =                             if      loader
4378  =                             ret
4379  =                             endif
4380  =                       ;
4381  =                             if      not loader
4382  =                       ;
4383  =210C E896FE      1FA5        CALL    parsave33r
4384  =210F E8F2F9      1B04        CALL    rename
4385  =2112 E994F8      19A9        jmp     cpydirloc
4386  =                             endif
4387  =                                                     ;ret ;jmp goback
4388  =                       ;
4389  =                       func24:
4390  =                       ;
4391  =                             ;return the login vector
4392  =                       ;
4393  =2115 8B1EEA22                MOV     BX,dlog
4394  =2119 EB26        2141        jmps    sthlret
4395  =                       ;     ret ;jmp goback
4396  =                       ;
4397  =                       func25:
4398  =                       ;
4399  =                             ;return selected disk number
4400  =                       ;
4401  =211B A0A424                  MOV     AL,curdsk
4402  =                       to_staret:
4403  =211E E9D3F4      15F4        jmp     staret
4404  =                       ;     ret ;jmp goback
4405  =                       ;
4406  =                       func26:
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4407
4408  =                       ;
4409  =                               ;set the subsequent dma address to info
4410  =                       ;
4411  =                                                       ;was MOV BX,INFO
4412  =2121 8916A024                  MOV     dmaad,dx        ;dmaad = info
4413  =2125 E94AF7      1872          JMP     setdata         ;to data dma address
4414  =                                                       ;ret ;jmp goback
4415  =                       ;
4416  =                       func27:
4417  =                       ;
4418  =                               ;return the login vector address
4419  =                       ;
4420  =2128 8B1EF822                  MOV     BX,alloca
4421  =212C EB13        2141          jmps    sthlret
4422  =                       ;       ret ;jmp goback
4423  =                       ;
4424  =  17BA                 func28  EQU     setro
4425  =                       ;
4426  =                               ;write protect current disk
4427  =                       ;
4428  =                                                       ;ret ;jmp goback
4429  =                       ;
4430  =                       func29:
4431  =                       ;
4432  =                               ;return r/o bit vector
4433  =                       ;
4434  =212E 8B1EE822                  MOV     BX,rodsk
4435  =2132 EB0D        2141          jmps    sthlret
4436  =                       ;       ret ;jmp goback
4437  =                       ;
4438  =                       func30:
4439  =                       ;
4440  =                               ;set file indicators
4441  =                               if      loader
4442  =                               ret
4443  =                               endif
4444  =                       ;
4445  =                               if      not loader
4446  =                       ;
4447  =2134 E86EFE      1FA5          CALL    parsave33r
4448  =2137 E8F2F9      1B2C          CALL    indicators
4449  =213A E976CF8      19A9         jmp     cpydirloc       ;lret=dirloc
4450  =                               endif
4451  =                                                       ;ret ;jmp goback
4452  =                       ;
4453  =                       func31:
4454  =                       ;
4455  =                               ;return address of disk parameter block
4456  =                       ;
4457  =                               if      loader
4458  =                               ret
4459  =                               endif
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4460
4461  =                     ;
4462  =                           if      not loader
4463  =213D 8B1EF422              MOV     BX,dpbaddr
4464  =                           endif
4465  =                     sthlret:
4466  =2141 891E9824              MOV     aret,bx
4467  =2145 C3                    RET                             ;jmp goback
4468  =                     ;
4469  =                     func32:
4470  =                     ;
4471  =                           ;set user code
4472  =                           if      loader
4473  =                     ;
4474  =                           ret
4475  =                           endif
4476  =                           if      not loader
4477  =2146 A01023                MOV     AL,linfo
4478  =2149 3CFF                  CMP     AL,0ffh
4479  =214B 7505          2152    JNZ     setusrcode
4480  =                                                           ;interrogate user code instead
4481  =214D A0A524                MOV     AL,usrcode
4482  =2150 EBCC          211E    jmps    to_staret               ;lret=usrcode
4483  =                     ;       ret ;jmp goback
4484  =                     ;
4485  =                     ;
4486  =                     setusrcode:
4487  =2152 241F                  AND     AL,1fh
4488  =2154 A2A524                MOV     usrcode,AL
4489  =2157 C3                    RET                             ;jmp goback
4490  =                           endif
4491  =                     ;
4492  =                     ;
4493  =                     func33:
4494  =                     ;
4495  =                           ;random disk read operation
4496  =                           if      loader
4497  =                           ret
4498  =                           endif
4499  =                     ;
4500  =                           if      not loader
4501  =                     ;
4502  =2158 E84FFE        1FAA   CALL    parsave36r               ;save 36 bytes from user seg
4503  =215B E899FE        1FF7   call    check_fcb
4504  =215E E979FD        1EDA   jmp     rdread                   ;to perform the disk read
4505  =                           endif
4506  =                                                           ;ret ;jmp goback
4507  =                     ;
4508  =                     func34:
4509  =                     ;
4510  =                           ;random disk write operation
4511  =                           if      loader
4512  =                           ret
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4513
4514  =                             endif
4515  =                     ;
4516  =                     ;
4517  =                             if      not loader
4518  =2161 E846FE      1FAA       CALL    parsave36r
4519  =2164 E97EFD      1EE5       jmp     rdwrite             ;to perform the disk write
4520  =                                                        ;ret ;,jmp goback
4521  =                             endif
4522  =                     ;
4523  =                     ;
4524  =                     func35:
4525  =                     ;
4526  =                             ;return file size (0-65536)
4527  =                             if      loader
4528  =                             ret
4529  =                             endif
4530  =                     ;
4531  =                             if      not loader
4532  =2167 E840FE      1FAA       CALL    parsave36r
4533  =216A E983FD      1F20       jmp     getfilesize
4534  =                                                        ;ret ;,jmp goback
4535  =                             endif
4536  =                     ;
4537  =                     ;
4538  =                     func36:
4539  =                     ;
4540  =                             ;set random record
4541  =                     ;
4542  =                             if      loader
4543  =                             ret
4544  =                             endif
4545  =                             if      not loader
4546  =216D E899EA      0C09       call    parsave36
4547  =                     ;
4548  =2170 E9EDFD      1F60       jmp     setrandom
4549  =                                                        ;ret ;,jmp goback
4550  =                             endif
4551  =                     ;
4552  =                     ;
4553  =                     func37:
4554  =                     ;
4555  =                             if      loader
4556  =                             ret
4557  =                             endif
4558  =                     ;
4559  =                             if      not loader
4560  =2173 A19924                  mov     ax,info
4561  =2176 F7D0                    not     ax
4562  =2178 50                      push    ax
4563  =2179 BBEA22                  mov     bx,offset dlog
4564  =217C 2307                    and     ax,[bx]
4565  =217E 8907                    mov     [bx],ax
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4566
4567  =2180 58                          pop     ax
4568  =2181 BBE822                      mov     bx,offset rodsk
4569  =2184 2307                        and     ax,[bx]
4570  =2186 8907                        mov     [bx],ax
4571  =2188 C3                          RET
4572  =                                 endif
4573  =                           ;
4574  =                           ;
4575  =                           goback:
4576  =                           ;
4577  =                                 ;arrive here at end of processing to return to user
4578  =                           ;
4579  =2189 803E132300                  cmp     resel,0
4580  =218E 7424            21B4       JZ      retmon
4581  =                                                          ;reselection may have
4582  =                                                          ;taken place
4583  =2190 8B1E9924                    MOV     BX,info
4584  =2194 803E962360                  cmp     highext,zero_readro     ;user zero and read/only
4585  =2199 7504            219F       jnz     goback1
4586  =219B 804F0880                    or      f8[bx],interfc_bit      ;set interface attribute r/o
4587  =                           goback1:
4588  =219F C60700                      MOV     b[bx],0                 ;fcb(0)=0
4589  =21A2 A01523                      MOV     AL,fcbdsk
4590  =21A5 0AC0                        OR      AL,AL
4591  =21A7 740B            21B4       JZ      retmon
4592  =                                                          ;restore disk number
4593  =21A9 8807                        MOV     [bx],al                 ;fcb(0)=fcbdsk
4594  =21AB A01423                      MOV     AL,olddsk
4595  =21AE A21023                      MOV     linfo,AL
4596  =21B1 E8E9FD          1F9D       CALL    curselect
4597  =                           ;
4598  =                           ;       return from the disk monitor
4599  =                           retmon:
4600  =21B4 803E2723FF                  cmp     parcopfl,true           ;test if parameter block
4601  =21B9 7503            21BE       jnz     retm1                   ;to be copied back to user
4602  =21BB E853EA          0C11       call    parunsave
4603  =21BE A17B24          retm1:     mov     ax,aret
4604  =21C1 8BD8                        mov     bx,ax
4605  =                           retmonret:
4606  =21C3 C3                          RET
4607  =  15F7                 func38    EQU     funcret
4608  =  15F7                 func39    EQU     funcret
4609  =                           func40:
4610  =                                 if      loader
4611  =                                 ret
4612  =                                 endif
4613  =                           ;
4614  =                                 if      not loader
4615  =                           ;
4616  =                                 ;random disk write with zero fill of unallocated block
4617  =                           ;
4618  =21C4 E8E3FD          1FAA       call    parsave36r
```

CP/M 86 V.1.1 - Vol I
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # C81 - 162

```
4619
4620  =21C7 C6060F2302              MOV     seqio,2
4621  =21CC B100                    MOV     CL,false
4622  =21CE E863FC      1E34        CALL    rseek1
4623  =21D1 75F0        21C3        JNZ     retmonret
4624  =21D3 E94CFB      1D22        jmp    diskwite
4625  =                             endif
4626  =                       ;
4627  =                       func47:
4628  =                       ;       chain to program
4629  =                       ;       move 128 byte from user dma to CCP's command buffer
4630  =                               if      loader
4631  =                               ret
4632  =                               endif
4633  =                       ;
4634  =                               if      not loader
4635  =                       ;
4636  =21D6 1E                        push    ds
4637  =21D7 B97000                    mov     cx,128
4638  =21DA BF0B00                    mov     di,offset ccpbuf        ;CCP command buffer
4639  =21DD 8B36A024                  mov     si,dmaad                ;user's DMA address
4640  =21E1 8E1EA224                  mov     ds,dmabase              ;user's DMA base
4641  =                       func471:
4642  =21E5 AC                        lodsb
4643  =21E6 0AC0                      or      al,al
4644  =21E8 7403        21ED          jz      func472
4645  =21EA AA                        stosb
4646  =21EB E2F8        21E5          loop    func471
4647  =                       func472:
4648  =21ED B080                      mov     al,80h
4649  =21EF 2AC1                      sub     al,cl
4650  =21F1 A20A00                    mov     comlen,al
4651  =21F4 1F                        pop     ds                      ;restore ds for BDOS
4652  =21F5 E93AED      0F32          jmp     f0freea
4653  =                               endif
4654  =                       ;
4655  =                       ;
4656  =                       func48:
4657  =21F8 C3                        ret
4658  =                       ;
4659  =                       ;
4660  =                       ;func49:
4661  =                               if      loader
4662  =                               ret
4663  =                               endif
4664  =                               if      not loader
4665  =                       ;       mov     bx,offset dmaad
4666  =                       ;       jmp     sthlret
4667  =                       ;
4668  =                               endif
4669  =                       ;
4670  =                       ;
4671  =                       ;********************* end BDOS filesystem part 3 ****************
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4672
4673                    end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4674
4675 =                  eject ! include bdosdat.a86     ; data area
4676 =                  ;********************* BDOS data area **********************
4677 =                  ;
4678 =                  ;
4679 =  0000            dseg    cpmsegment
4680 =                  ;       Variables in data segment:
4681 =                          org     bdosdataseg
4682 =                  ;
4683 =                  ;     the following pointer to the BIOS is used by the BDOS
4684 =                  ;        and changed by the NET when present in memory
4685 =                  ;
4686 =2200 1316         biosco  dw      offset bioscl   ;offset of 'call BIOS' instruction
4687 =2202 0000         bioscs  dw      00h             ;segment location of BIOS
4688 =  2200            biosptr equ     biosco
4689 =                  ;
4690 =                  ;
4691 =2204 A40B         pererr  dw      persub          ;permanent error subroutine
4692 =2206 B00B         seler   dw      selsub          ;select error subroutine
4693 =2208 B50B         roderr  dw      rodsub          ;ro disk error subroutine
4694 =220A BA0B         rofer   dw      rofsub          ;rofile error subroutine
4695 =                  ;
4696 =                  ;
4697 =                  ;
4698 =                  ;
4699 =                          if      not loader
4700 =220C               header  rs      128
4701 =228C                      rs      72
4702 =  2254            ccpbuff equ     header+72
4703 =  220F            abs8080 equ     word ptr header+3
4704 =  2211            min8080 equ     word ptr header+5
4705 =  2213            max8080 equ     word ptr header+7
4706 =                          endif
4707 =                  ;
4708 =                  ;
4709 =                                                  ;returned in bx
4710 =22D4 0000         pag0    dw      0               ;address of user's page zero
4711 =22D6 00           ip0     db      0               ;initial page value for IP register
4712 =                  ;
4713 =                  ;
4714 =                  ;
4715 =                  ;
4716 =                  ;     memory control block
4717 =                  ;
4718 =22D7 0000         umembase        dw      0       ;user'sbase for memory request
4719 =22D9 0000         umemlg  dw      0               ;length of memory req
4720 =22DB 00           contf   db      0               ;flag indicates added memory is avail
4721 =                  ;
4722 =                  ;
4723 =22DC 0000         hold_info       dw      0       ;save info
4724 =22DE 0000         hold_spsave     dw      0       ;save user SP during program load
4725 =22E0 0000         hold_sssave     dw      0       ;save user SS during program load
4726 =
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        ;
=22E2 00                mod8080         db      0
=                       ;
=                       ;
=                       ;       byte I/O variables:
=                       ;
=22E3 00                compcol db      0               ;true if computing column position
=22E4 00                strtcol db      0               ;starting column position after read
=22E5 00                column  db      0               ;column position
=22E6 00                kbchar  db      0               ;initial key char = 00
=                       ;
=                       ;       common values shared between bdosi and bdos
=                       ;
=                       ;       data areas for file system:
=                       ;
=                       ;       initialized data
=                       ;
=22E7 E5                efcb    db      empty           ;0e5=available dir entry
=22E8 0000              rodsk   dw      0               ;read only disk vector
=22EA 0000              dlog    dw      0               ;logged-in disks
=                       ;
=                       ;       curtrka - alloca are set upon disk select
=                       ;       (data must be adjacent, do not insert variables)
=                       ;       (address of translate vector, not used)
=                       ;
=22EC 0000              cdrmaxa dw      0               ;pointer to cur dir max value
=22EE 0000              curtrka dw      0               ;current track address
=22F0 0000              curreca dw      0               ;current record address
=22F2 0000              buffa   dw      0               ;pointer to directory dma address
=22F4 0000              dpbaddr dw      0               ;current disk parameter block address
=22F6 0000              checka  dw      0               ;current checksum vector address
=22F8 0000              alloca  dw      0               ;current allocation vector address
= 0008                  addlist equ     8               ;'$-buffa' = address list size
=                       ;
=                       ;       sectpt - offset obtained from disk parm block at dpbaddr
=                       ;       (data must be adjacent, do not insert variables)
=22FA 0000              sectpt  dw      0               ;sectors per track
=22FC 00                blkshf  db      0               ;block shift factor
=22FD 00                blkmsk  db      0               ;block mask
=22FE 00                extmsk  db      0               ;extent mask
=22FF 0000              maxall  dw      0               ;maximum allocation number
=2301 0000              dirmax  dw      0               ;largest directory number
=2303 0000              dirblk  dw      0               ;reserved allocation bits for direct.
=2305 0000              chksiz  dw      0               ;size of checksum vector
=2307 0000              offsetv dw      0               ;offset tracks at beginning
=2309                   endlist rs      0               ;end of list
= 000F                  dpblist equ     (offset endlist)-(offset sectpt)        ;size
=                       ;
=                       ;       local variables
=2309 0000              tranv   dw      0               ;address of translate vector
=230B 00                fcbcopied db    0               ;set true if copy$fcb called
=230C 00                rmf     db      0               ;read mode flag for open$reel
=230D 00                wflag   db      0               ;write flag sent to BIOS for blocking
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
4780
4781  =230E 00              dirloc  db      0               ;directory flag in rename, etc.
4782  =230F 00              seqio   db      0               ;1 if sequential i/o
4783  =2310 00              linfo   db      0               ;low(info)
4784  =2311 00              dminx   db      0               ;local for diskwrite
4785  =2312 00              single  db      0               ;set true if single byte
4786  =                                                     ;allocation map
4787  =2313 00              resel   db      0               ;reselection flag
4788  =2314 00              olddsk  db      0               ;disk on entry to bdos
4789  =2315 00              fcbdsk  db      0               ;disk named in fcb
4790  =2316 00              rcount  db      0               ;record count in current fcb
4791  =2317 00              extval  db      0               ;extent number and extmsk
4792  =2318 0000            vrecord dw      0               ;current virtual record
4793  =  2318               lvrecord equ    byte ptr vrecord ;low (vrecord)
4794  =231A 0000            arecord dw      0               ;current actual record
4795  =231C 0000            ablock  dw      0               ;current actual block# * blkmsk
4796  =                     ;
4797  =                     ;       local variables for directory access
4798  =231E 00              dptr    db      0               ;directory pointer 0,1,2,3
4799  =231F 0000            drec    dw      0               ;directory record 0,1,...,dirmax/4
4800  =                     ;
4801  =                     ;       Special 8086 variables:
4802  =                     ;
4803  =2321 0000            srcha   dw      0               ;search address
4804  =2323 0000            user_parm_seg   dw      0       ;holds user parameter seg during load
4805  =2325 0000            retrn_es        dw      0       ;ES register returned to user
4806  =2327 00              parcopfl        db      0       ;true if parameter block copied
4807  =2328 00              parlg           db      0       ;length of parameter block
4808  =                             if      not loader
4809  =2329 00              nallocmem       db      0       ;no. of allocated memory segments
4810  =232A 00              ncrmem          db      0       ;no. of available memory segments
4811  =232B 00000000        crmem           dw      0,0     ;memory table (16 elements)
4812  =232F 000000000000                    dw      0,0,0,0,0,0
4813        000000000000
4814  =233B 000000000000                    dw      0,0,0,0,0,0,0,0
4815        000000000000
4816        00000000
4817  =234B 000000000000                    dw      0,0,0,0,0,0,0,0
4818        000000000000
4819        00000000
4820  =235B 000000000000                    dw      0,0,0,0,0,0,0,0
4821        000000000000
4822        00000000
4823  =                     ;
4824  =  0028               mem_stack_length        equ     40
4825  =236B                 memstack        rs      mem_stack_length
4826  =                                             ;8 possible allocations
4827  =  0000               stbase  equ     word ptr 0
4828  =  0002               stlen   equ     word ptr 2
4829  =  0004               ccpflag equ     byte ptr 4
4830  =2393 00              nccpalloc       db      0       ;number of current ccp allocations
4831  =2394 0000            mem_stk_ptr     dw      0       ;current memory stack location
4832  =                     ;
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4833
4834 =                          endif
4835 =                      ;
4836 =                      ;   added for user zero search
4837 =2396 00               high_ext        db      0       ;fcb high extend bits
4838 =2397 00               search_user0    db      0       ;search user 0 on 'open'
4839 =2398 0000             xdcnt   dw      0               ;dcnt saved on user 0 search
4840 =  2399                found_in_zero   equ     xdcnt+1 ;found file in user zero
4841 =239A 00               user_zero_pass  db      0       ;search user zero flag
4842 =                      ;
4843 =                      ;
4844 =                      ;
4845 =                      ;       error messages
4846 =                      ;
4847 =239B 42646F732045     dskmsg  db      'Bdos Err On '
4848        7272204F6E20
4849 =23A7 203A2024         dskerr  db      ' : $'
4850 =23AB 426164205365     permsg  db      'Bad Sector$'
4851        63746F7224
4852 =23B6 53656C656374     selmsg  db      'Select$'
4853        24
4854 =23BD 46696C6520       rofmsg  db      'File '
4855 =23C2 522F4F24         rodmsg  db      'R/O$'
4856 =                      ;
4857 =                      ;       Local buffer area:
4858 =                      ;
4859 =23C6                  loc_par_area    rb      40      ;local user parameter (FCB,...)
4860 =23EE                  stackarea       rw      ssize   ;stack size
4861 =248E                  endstack        rb      0       ;top of stack
4862 =                      ;
4863 =                      ;********************* end BDOS data area **********************
4864 =                      ;
4865 =                      ;
4866 =                      ;    SYSTEM CONTROL BLOCK....Address returned by function 49
4867 =                      ;
4868 =                              org     biosoffset-6ch
4869 =2494 0000             SS_save         dw      0
4870 =2496 0000             SP_save         dw      0
4871 =2498 00               stack_flag      db      0h
4872 =2499 0000             info    dw      0                       ;information address
4873 =249B 0000             aret    dw      0                       ;address value to return
4874 =  249B                lret    EQU     byte ptr aret           ;low(aret)
4875 =                      ;
4876 =249D 0000             parametersegment dw     0       ;user parameter segment
4877 =                      ;
4878 =                              org     biosoffset-60h
4879 =                      ;
4880 =24A0 0000             dmaad   dw      0               ;user DMA address
4881 =24A2 0000             dmabase dw      0               ;user DMA base
4882 =24A4 00               curdsk  db      0               ;current user disk
4883 =24A5 00               usrcode db      0               ;current user number
4884 =24A6 0000             dcnt    dw      0               ;directory index
4885 =  24A6                ldcnt   equ     byte ptr dcnt   ;low(dcnt)
```

;filled in by config
COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
4886
4887  =24A8 00              srchl   db      0               ;search length
4888  =24A9 0000            infosave        dw      0       ;save for FCB address
4889  =24AB 0000            srcha_seg       dw      0       ;DS segment from last search
4890  =24AD 0000            cbuffa          dw      0       ;address of directory buffer
4891  =                     ;
4892  =                     ;
4893  =24AF 00              background      db      0       ;background despool checks for abort
4894  =                                                     ;01h...despool in memory
4895  =                                                     ;03h....abort despool
4896  =24B0 00              desplchars      db      0       ;number of characters for despool
4897  =24B1 00              mxdos           db      0       ;BDOS diskio flag
4898  =24B2 00              subflag db      0               ;submit flag
4899  =24B3 00              buf_flg db      0               ;xsub flag if buffer not empty
4900  =24B4 0000            ercode  dw      0               ;error return code for submit
4901  =                     ;
4902  =24B6 00              listcp  db      0               ;listing toggle...set by ctl-p
4903  =24B7 00              curdrvs db      0               ;current drives to search
4904  =24B8 00                      db      0
4905  =24B9 00                      db      0
4906  =24BA 00              intflag db      0
4907  =24BB 0000            roldate dw      0               ;rolander date
4908  =24BD                 roltime rs      3               ;rolander time
4909  =24C0                 asdate  rs      8               ;ascii date
4910  =24C8 00                      db      0               ;end string
4911  =24C9                 astime  rs      9               ;ascii time
4912  =24D2                         rs      14
4913  =24E0 50              colwdth db      80              ;screen column width
4914  =24E1 00                      db      0               ;end string
4915  =24E2                         rs      24              ;expansion space
4916  =24FA 00                      db      0               ;end of this data area
4917  =                     ;
4918  =                     ;
4919                                end
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
4920
4921                        eject
4922                        ;********************** end of BDOS **********************
4923                        end
4924
4925
4926  END OF ASSEMBLY.  NUMBER OF ERRORS:  0.  USE FACTOR: 57%
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
ABLOCK          231C V  2472  3773  4795#
ABS             0003 N   476#  556   559   730   732   818   871
ABSB080         220F V   716  4703#
ACTUAL          0001 N   475#  821
ADDH            17F6 L  2600#
ADDLIST         0008 N  2317  4759#
ADJUST1         10D1 L  1189# 1209
ADJUSTSTACK     10C8 L  1114  1184#
ALLA1           1281 L  1487  1499#
ALLA2           128B L  1484  1504#
ALLA3           129C L  1507  1518#
ALLAEX          1283 L  1495  1502  1514  1523#
ALLDS           0D87 L   667   677   679#
ALLDS1          0D84 L   674   673#
ALLES           0D8E L   660   683#
ALLMODULES      20C8 L  4286  4288#
ALLOC1          117E L  1285  1294#
ALLOCA          22F8 V  2823  2930  2940  4420  4758#
ALLOCEXIT       1185 L  1292  1298#
ALLOCMEM        0037 N   459#  607   616
ALLOCMEMABS     0038 N   460#  568   584
ALLOCMEMAT      1261 L  1071  1472#
ALLOCSEG        116C L  1051  1275#
ALLOCT          1722 L  2461# 3653  3692
ALLOCWD         1D87 L  3731  3736#
ARECORD         231A V  2352  2376  2385  2458  2463  2470  2476  3724  3765  3774
                        3778  3786  4794#
ARET            249B V   328   847   904  4466  4603  4873# 4874
ASDATE          24C0 V  4909#
ASTIME          24C9 V  4911#
ATRAN           1729 L  2467# 3657  3754
B               0000 N    51# 1631  1801  1805  1815  1822  2002  2328  2333  2620
                        2934  2960  3117  3138  3162  3566  3800  3842  3958  4157
                        4165  4234  4281  4285  4588
BACKGROUND      24AF V  1748  4893#
BACKSP          1569 L  2055# 2059
BACKSP1         1572 L  2049  2060#
BACKUP          1475 L  1856# 2003  2057
BACKX           151D L  1995# 2004
BADSEEK         1EC9 L  3920  3938  3944  3950#
BDOSCODESIZE    1700 N    28#   29
BDOSDATASEG     2200 N    29# 4681
BDOSE           0B06 L   263#
BDOSE0          0B34 L   283   289#
BDOSE1          0B47 L   294   296#
BDOSINT         00E0 N   173#  514
BDOSIOFFSET     3000 N    25#
BDOSOFFSET      0B00 N    26#   29   199   253
BIOS            2500 L   82#   83    85    86    87    88    89    90    91    92
                         93    94    95    96    97    98    99   100   101   102
                        103   104
BIOSCL          1613 L  2224# 4636
BIOSCO          2200 V  4686# 4688
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
BIOSCONT          0F68 L   952#
BIOSCS            2202 V  4182  4184  4687#
BIOSOFFSET        2500 N    27#   81   957  4868  4878
BIOSPTR           2200 V  2214  4688#
BLKMSK            22FD V  2474  3781  4765#
BLKSHF            22FC V  2407  2469  4764#
BLOCKOK           1D68 L  3717  3722#
BOOTF             2500 L    33#
BRANCH            0B97 L   336   340   343#
BUFFA             22F2 V  2313  2316  2530  2598  2725  2736  3766  4755#
BUFFLG            24B3 V  4899#
BUFFNZERO         1BE5 L  3422  3428#
CALLOPEN          203F L  4208  4212#
CBUFFA            24AD V  2314  4890#
CCPBUF            000B L    69#  4638
CCPBUFF           2254 V   752   807  4702#
CCPENT            0000 L    64#
CCPFLAG           0004 N   627   994  1231  1232  1248  4829#
CCPOFFSET         0000 N    24#   63    66    68    70
CDRMAXA           22EC V  2303  2566  2637  2946  4752#
CHECKA            22F6 V  2678  4757#
CHECKFCB          1FF7 L  4161#  4339  4503
CHECKSUM          1831 L  2665#  2797
CHECKWRITE        17E4 L  2586#  3149  3263  3501  3678
CHKSIZ            2305 V  2670  4770#
CKRODIR           17D1 L  2570#  3160  3231
CKROFILE          17D4 L  2574#  3680
CLOSE             1B99 L  3378#  3538  3914  4268
CLRMODNUM         1805 L  2617#  4206  4287  4369
CMPECO            179A L  2532#  2535
CMPECS            1791 L  2527#  2676
CMPERR            1EF0 L  3990#  4043  4071
COLUMN            22E5 V  1739  1798  1850  1877  1913  1948  1997  2052  4735#
COLWDTH           24E0 V  4913#
COMLEN            000A V    67#   922  1753  4650
COMPCDR           1814 L  2634#  2645  2687  3052
COMPCOL           22E3 V  1775  1949  2047  2053  2058  4733#
COMPEXT           19AF L  2991#  3035
COMPOUT           142A L  1778  1795#
COMPUTEENDSEG     1212 L  1406#  1425  1428
COMPUTEENDSEGMEN  1218 L  1414#  1445  1481  1592  1629  1656
CONBO             1407 L  1727  1783#
CONB1             140A L  1717  1766#
CONBRK            1388 L  1714#  1783  2179
CONECH            1395 L  1688#  2106
CONIN             1386 L   389  1677#  1690  1928
CONINF            2509 L    87#  2200
CONOUF            250C L    88#  2210
CONOUT            140D L  1771#  1835  1844  1849  1872  1881  1888  1890  2101
CONSTF            2506 L    86#  2206
CONTF             22DB V  4720#
COPYDIR           1AEE L  3237#  3284  3308
COPYFCB           1AEA L  3230#  3526
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
CPMSEGMENT      0000 N    23#   59   198  4679
CPYDIRLOC       19A9 L  2984#  4329  4385  4449
CR              000D N   119#  1703  1887  1932  2098
CRLF            149A L   381  1873  1885#  1977
CRLFP           1482 L  1868#  2011  2023
CRLFP0          148A L  1876#  1882
CRMEM           232B V  1261  1468  1585  1621  1647  4811#
CTL             005E N   121#  1834
CTLC            0003 N   108#   354  1730  2077
CTLCCHECK       13D7 L  173   1737#  2084
CTLCCHECK0      13EE L  1741  1747#
CTLCCHECK1      1404 L  1749  1751  1759#
CTLE            0005 N   109#  1971
CTLH            0008 N   110#  1709  1812  1864  1939
CTLOUT          1452 L  1825#  2038  2073
CTLP            0010 N   111#  1983
CTLR            0012 N   112#  2016
CTLS            0013 N   113#  1726
CTLU            0015 N   114#  2008
CTLX            0018 N   115#  1991
CTLZ            001A N   116#
CURDRVS         24B7 V  4189  4903#
CURDSK          24A4 V   382  2293  2541  2552  4088  4113  4146  4190  4401  4882#
CURRECA         22F0 V  2307  4754#
CURSELECT       1F9D L  4111#  4152  4196  4596
CURTRKA         22EE V  2305  2345  2390  4753#
DCNT            24A6 V  2373  2636  2749  2761  2768  2770  3140  3382  4230  4884#
                        4885
DELET           1A8D L  3147#  4328
DELET0          1A95 L  3152#  3167
DESPLCHARS      24B0 V  4896#
DIOCOMP         16A0 L  2354  2365#
DIRBLK          2303 V  2939  4769#
DIRINP          15C4 L  2131  2136#
DIRLOC          230E V  2987  3012  3105  4781#
DIRMAX          2301 V  2564  2767  4768#
DIRREC          0004 N   135#   138
DIRTOUSER       188B L  2733#  4314
DISKEOF         1D1A L  3637  3645  3654  3662#
DISKF           000C N   209#
DISKIO          000D N    75#   273
DISKREAD        1CDD L  3623#  3976
DISKWITE        1D22 L  3673#  3986  4624
DIVMEM          0D43 L   620   648#
DIVMEM1         0D55 L   650   656#
DLOG            22EA V  4087  4105  4107  4188  4393  4563  4746#
DMAAD           24A0 V   246   976  2718  2737  4412  4639  4880#
DMABASE         24A2 V   965   974  2716  4640  4881#
DMINX           2311 V  3703  3729  4784#
DMPOSITION      16E1 L  2405#  2454  3702
DMSET           1C01 L  3431  3449#
DPBADDR         22F4 V  2321  4463  4756#
DPBLIST         000F N  2323  4773#
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
DPTR              231E V  2599  2790  4798#
DREC              231F V  2377  2669  2677  4799#
DSKERR            23A7 V   384  4849#
DSKMAP            0010 N   150# 2438  2874  2876  3270  3282  3396  3403  3727
DSKMSG            239B V   385  4847#
DSKMSK            0003 N   138# 2783  3107
DSKSHF            0002 N   136# 2374
DSKW11            1DEB L  3756  3761  3788#
DSKWR0            1D3E L  3687  3689#
DSKWR1            1D90 L  3695  3745#
DSKWR10           1D98 L  3750  3752#
DSKWR2            1E01 L  3797  3803#
DSKWR3            1E2C L  3819  3825  3843#
DSKWRU            1D8E L  3735  3742#
DVERS             0022 N    40#  241
ECHOC             13A5 L  1691  1700# 1828
EFCB              22E7 V  3503  4744#
EMPTY             00E5 N   131# 2960  3047  3162  4744
ENDADJUSTSTACK    1130 L  1205  1224  1237#
ENDDIR            FFFF N   127# 2760
ENDFREEUSER       1149 L  1246  1249  1252#
ENDLIST           2309 V  4772# 4773
ENDMERGE          1C28 L  3470  3485#
ENDOFDIR          139F L  2743# 2955  3034  3154  3275  3302  3316  3392  3506  3541
                          3577  3587  4038  4244
ENDSEARCH         1A44 L  3062  3103#
ENDSTACK          248E V   287  4861#
EOFRE             14D5 L  1934# 1938
ERCODE            24B4 V  4900#
ERRFLG            0BC3 L   353   373   377#
EXTMSK            22FE V  2505  2995  3554  4766#
EXTNUM            000C N   141# 2433  3073  3150  3265  3283  3299  3307  3336  3458
                          3475  3545  3565  3895  3921  3998  4024
EXTVAL            2317 V  2415  2507  4791#
FOFREE            0F2D L   375   921# 1760
FOFREE1           0F3E L   925   930#
FOFREEA           0F32 L   923# 4652
F8                0008 N   161# 4128  4131  4586
FALSE             0000 N    45#   47   319   554  2333  3030  3032  3093  3675  3982
                          4222  4621
FCBCOPIED         230B V  3486  3537  3557  4777#
FCBDSK            2315 V   331  4149  4589  4789#
FCBLEN            0020 N   134#  135   151   152  2876  3233  3403  3458  3514
FCBNZERO          1BDB L  3417  3420#
FCBSHF            0005 N   139# 2785
FILL1             1DC4 L  3774# 3785
FILPAG1           0EBB L   861#  892
FILPAG2           0EC1 L   864#  868
FILPAG3           0ECC L   866   870#
FILPAG4           0EDF L   875   878#
FILPAG5           0EFC L   969   883   888#
FINDSEG           0D2E L   632#  658   665   685
FINDSEG1          0D34 L   635#  639
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
FOUNDINZERO     2399 V  4209  4223  4840#
FREE            1031 L   933  1086# 1250
FREE01          1052 L  1093  1101  1104#
FREE1           107F L  1126  1136#
FREE2           1084 L  1120  1139#
FREE3           1095 L  1140  1153#
FREEUCER        1133 L   931  1102  1244# 1251
FUN53B          0FC8 L  1011  1013#
FUN54C          0FF2 L  1037  1039#
FUNC0           0F26 L   203   915#
FUNC1           15AB L   203  2103#
FUNC10          15EA L   207  2170#
FUNC11          15F1 L   207  2176#
FUNC12          0AEC L   210   240#
FUNC13          2006 L   210  1742  4178#
FUNC13A         2012 L  4183  4185#
FUNC14          1F9D L   210  4196#
FUNC15          2027 L   210  4201#
FUNC16          20A1 L   212  4263#
FUNC17          20A7 L   212  4271#
FUNC18          20D2 L   212  4296#
FUNC19          20EB L   212  4318#
FUNC2           1463 L   203  2109#
FUNC20          20F4 L   214  4334#
FUNC21          20FD L   214  4343#
FUNC22          2103 L   214  4360#
FUNC23          210C L   214  4374#
FUNC24          2115 L   216  4389#
FUNC25          211B L   216  4397#
FUNC26          2121 L   216  4406#
FUNC27          2128 L   216  4416#
FUNC28          17BA L   218  4424#
FUNC29          212E L   218  4430#
FUNC3           15B0 L   203  2113#
FUNC30          2134 L   218  4438#
FUNC31          213D L   218  4453#
FUNC32          2146 L   220  4469#
FUNC33          2158 L   220  4493#
FUNC34          2161 L   220  4508#
FUNC35          2167 L   220  4524#
FUNC36          216D L   222  4538#
FUNC37          2173 L   222  4553#
FUNC38          15F7 L   222  4607#
FUNC39          15F7 L   222  4608#
FUNC40          21C4 L   224  4609#
FUNC47          21D6 L   225  4627#
FUNC471         21E5 L  4641# 4646
FUNC472         21ED L  4644  4647#
FUNC48          21F8 L   226  4656#
FUNC49          0AF1 L   227   245#
FUNC50          0F57 L   228   942#
FUNC51          0F76 L   228   961#
FUNC52          0F80 L   228   971#
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
FUNC53          0FAC L   228   998#
FUNC53A         0FAF L  1001#
FUNC53B         0FB1 L  1003#
FUNC54          0FCE L   230  1019#
FUNC54A         0FD1 L  1022#
FUNC54B         0FD3 L  1024# 1032
FUNC55          0FF4 L   230  1043#
FUNC55A         0FF7 L  1046#
FUNC56          1006 L   230  1056#
FUNC56A         1009 L  1059#
FUNC57          1044 L   230  1095#
FUNC58          10A4 L   232   926  1160#
FUNC59          0E1D L   232   777#
FUNC6           15B5 L   205  2123#
FUNC7           15D3 L   205  2146#
FUNC8           15D8 L   207  2153#
FUNC9           15DE L   207  2160#
FUNCRET         15F7 L  2184# 4607  4608
FUNCTAB         0A80 V   203#  347
FWFMSK          0080 N   146#  2627  3388  3811
GET16           0DF1 L   743#  826
GET16A          0E04 L   745   754#
GETALLOCBIT     18E5 L  2802# 2841  3188  3201
GETBLOCK        1AB0 L  3171# 3714
GETDM           16F9 L  2434# 2457  3710
GETDMD          170E L  2441  2446#
GETDPTRA        17EF L  2572  2595# 2873  2959  3057  3161  3244  3326  3397  4041
GETEXTA         1745 L  2480# 2504  3322  4284
GETFCB          175C L  2495# 3597  3628  3681
GETFILESIZE     1F20 L  4022# 4533
GETIOBF         2539 L   103# 2150
GETMA1          100B L  1061# 1070
GETMAX1         11AE L  1338# 1346  1350
GETMAXABS       0036 N   458#  577
GETMAXEX        11C8 L  1342  1352#
GETMAXFIN       11D4 L  1357  1359#
GETMAXMEM       0035 N   457#  612
GETMAXSEG       11A8 L  1006  1328#
GETMLKENTRY     114A L  1000  1021  1045  1058  1099  1255#
GETMEMENTRY1    1153 L  1092  1259#
GETMEMERR       102B L  1005  1008  1023  1028  1047  1050  1060  1065  1080# 1106
                        1110  1112  1116
GETMEMEX        102D L  1079  1082#
GETMEMOK        1021 L  1017  1040  1053  1074# 1103  1132  1137  1149
GETMEMOK1       1027 L  1077#
GETMEMORY       0C4B L   519#  570   578   585   608   613   617
GETMENTEX       1169 L  1266  1268  1270#
GETMIN1         11E0 L  1375# 1384  1386  1391
GETMINEX        1201 L  1379  1393#
GETMINFIN       120D L  1383  1398  1400#
GETMINFIRST     0C54 L   526#  668
GETMINSEG       11D9 L  1049  1366#
GETMODNUM       17FB L  2608# 2619  2624  3387  3807  3957
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
GETNEXTSIZE       1F59 L  4055  4060#
GETSEGTABF        2536 L   102# 1168  1552
GETSIZE           1F37 L  4037# 4062
GETUMEM           0C6B L   542# 800
GETUMEMOK         0D17 L   605  610   621#
GMF1              0C68 L   534  536#
GOBACK            2189 L   333  4575#
GOBACK1           219F L  4585  4587#
GOERR             1619 L  2266# 2369  2592
GTFCBA            174F L  2487# 2498  2512  3639
HEADER            220C V   548  633   721   790   812   862  4700# 4702  4703  4704
                         4705
HIGHEXT           2396 V  4127  4132  4162  4231  4247  4255  4584  4837#
HOLDINFO          22DC V   503  782   895  4723#
HOLDSPSAVE        22DE V   786  900  4724#
HOLDSSSAVE        22E0 V   784  898  4725#
HOME              1604 L  2340# 2943  3021
HOMEF             2518 L    92# 2342
INBDOS            0B19 L   271  275#
INDEX             1713 L  2452# 3648  3691
INDICO            1B31 L  3301# 3310
INDICATORS        1B2C L  3297# 4448
INFO              2499 V   325  402   405   440   781   896  1258  1915  2436  2482
                         2490  2612  3009  3242  3268  3328  3361  3400  3502  3503
                         3507  3510  3544  3679  3726  3855  4027  4069  4126  4135
                         4156  4164  4233  4251  4280  4309  4560  4583  4872#
INFOSAVE          24A9 V   403  439  4308  4883#
INIT0             195C L  2933# 2936
INIT2             197A L  2952# 2961  2982
INITCS            185C L  2681  2691#
INITIALIZE        194D L  2921# 4108
INTERFCBIT        0080 N   162# 4128  4586
INTERRUPTBIT      0200 N   174#  273
INTFLAG           24BA V  4906#
INVIS             000A N   157#
IPO               22D6 V   709  715   855  4711#
JGETMEMOK         1093 L  1148# 1158
JMPERR            0CE0 L   582  596#  615
JMPFOFREE         0BC0 L   356  374#
JMPGOERR          17EC L  2582  2591#
JMPSRCHFIN        19EA L  3037# 3054
JOINSMENSEGMENTS  12BA L  1136  1534#
JPMEMERR          0FB8 L  1002  1007#
KBCHAR            22E6 V  1679  1715  1765  4736#
L11               1429 L  1791  1793#
L19               14D8 L  1933  1935#
L23               1597 L  2081  2085#
L24               159E L  2090  2092#
L26               15CE L  2139  2141#
L41               1922 L  2881  2883#
L42               1948 L  2908  2912#
L51               1B1D L  3276  3278#
L65               1EE4 L  3975  3977#
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
L66             1EEF L  3985  3987#
L67             1F8B L  4096  4098#
LAB1            133B L  1623#  1632
LABEX           1354 L  1633  1635#
LABNOT          1352 L  1627  1634#
LDCNT           24A6 V  2782  3106  4885#
LEFTTST         1AB2 L  3179#  3206  3223
LEN             0000 N   482#   881   886
LENH            0002 N   483#   880   835
LF              000A N   120#  1705  1819  1889  1937
LINELEN         153F L  1952  2018#
LINFO           2310 V   326   916  4112  4140  4477  4595  4783#
LISTCP          24B6 V  1790  1986  4902#
LISTF           250F L    89#  2196
LISTFILE        0005 N    74#   270
LISTSTF         252D L     %
LLAB1           1366 L  1649#  1659
LLABEX          137F L  1660  1663#
LLABNOT         137D L  1654  1661#
LOAD1           0E51 L   795   799#
LOAD2           0E58 L   803#
LOADER          0000 N    47#   189  1738  2213  2216  2801  2835  2909  3145  3170
                        3254  3291  3496  3504  3667  4299  4304  4321  4325  4344
                        4349  4363  4367  4377  4381  4441  4445  4457  4462  4472
                        4476  4496  4500  4511  4517  4527  4531  4542  4545  4555
                        4559  4610  4614  4630  4634  4661  4664  4699  4808
LOADERR         0F1E L   797   903#
LOADERRJMP      0E4E L   796#   802
LOADNO          0E6C L   813#   842
LOADN1          0E78 L   820#
LOADN2          0E7D L   824#   837
LOADN3          0E93 L   831   835#
LOADOK          0F06 L   887   894#  905
LOCPARAREA      23C6 V   404   438  4859#
LRET            249B V   573   609   794  1003  1743  2183  2975  3108  3381  3492
                        3644  3749  3833  3918  3928  3942  3963  4874#
LSTFCB          001F N   151#
LSTREC          007F N   132#  3683  3818
LVRECORD        2318 V  2409  2473  2500  2519  3629  3682  3794  3840  4793#
MAKE            1C35 L  3497#  3585  3940  4370
MAKE0           1C5D L  3516#  3520
MAX             0007 N   478#   529   531   561   564   583   589   591   659   666
                        673   675   678   681   687   695   697   874   877
MAX3080         2213 V   654  4705#
MAXALL          22FF V  2326  2906  2925  3195  4767#
MAXEXT          001F N   142#  3002  3548
MAXMOD          000F N   145#  3570
MEMBASE         0000 N   167#   990  1014  1015  1034  1035  1144  1145  1154  1155
                        1190  1289  1290  1296  1297  1410  1446  1447  1482  1483
                        1499  1511  1512  1518  1519  1524  1593  1594  1597  1630
                        1658
MEMDESCRLG      0004 N   169#  1025  1062  1076  1100  1261  1321  1339  1376  1465
                        1555  1559  1585  1587  1621  1624  1647  1650
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
MEMLG            0002 N  163#  992  1009  1010  1012  1033  1036  1038  1105  1130
                         1131  1146  1147  1156  1157  1199  1203  1215  1222  1267
                         1283  1284  1295  1344  1381  1382  1409  1500  1501  1513
                         1520  1527  1540  1541  1542
MEMSTACK         236B V  1166  1188  4825#
MEMSTACKLENGTH   0028 N  1188  4824# 4825
MEMSTKPTR        2394 V   622   932   938   989  1167  1236  1247  4831#
MERGEO           1BC9 L  3404# 3453
MERGED           1BEB L  3406  3432#
MERGERR          1C30 L  3430  3443  3490#
MERGEZERO        1B8E L  3367# 3434  3436
MIN              0005 N   477#  530   532   562   581   588   600   672   680   686
                          700   873
MIN3080          2211 V   651  4704#
MOD8080          22E2 V   545   554   649   713   882  4728#
MODNUM           000E N   144# 2613  3565  3901  3924  4006
MOVE             161B L  2274# 2318  2324  2741  3246  3331
MXDOS            24B1 V   276   319  4897#
NALLOCMEM        2329 V   995  1163  1185  1235  1245  1264  4809#
NAMLEN           000F N   148# 3314  3390  3512  3514  3575  4235  4290
NBIOSFUNCS       0015 N   105#
NCCPALLOC        2393 V   628   924   934  1165  1740  1750  4830#
NCRMEM           232A V  1004  1027  1064  1075  1176  1311  1313  1341  1378  1463
                         1464  1589  1626  1653  4810#
NEW              124F L  1153  1461# 1522
NEWALLOCATION    0F0F L   987# 1052  1072
NEWCHECKSUM      182F L  2662# 2699
NEXLOAD          0E96 L   817   822   839#
NEXTSTACK        10F0 L  1193  1201  1207#
NFUNCS           0036 N   237#  338
NFUNCS80         0029 N   235#  335   337
NOMAX            0CE2 L   590   599#
NOPBLOCK         1D5C L  3706  3712#
NOSELECT         1FED L  4143  4153#
NOSPACE          1E29 L  3836  3841#
NOTABS           0CCD L   557   587#
NOTBACKSP        144A L  1813  1817#
NOTC             1597 L  2079  2087#
NOTCCP           0D2C L   626   629#
NOTDS            0C89 L   553   555#
NOTE             150E L  1972  1981#
NOTFOUND         00FE N   160# 4209  4223
NOTH             14EE L  1940  1953#
NOTINTERFC8IT    007F N   163# 4131
NOTP             1518 L  1984  1988#
NOTR             1574 L  2017  2062#
NOTRUB           14FE L  1956  1969#
NOTU             153B L  2009  2014#
NOTX             1530 L  1992  2005#
NOUPDATE         1E0A L  3815#
NOWRITE          17AE L  2548# 2588  3383
NXTBAS1          0DA4 L   696   699#
NXTBASE          0D9A L   693#  737
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
NXTREC          0020 N   152#  153  2491  3329  3888  4070
NXTSEG          0CE5 L   551   574   586   592   594   601#
OFF             0000 N    43#   46
OFFSETV         2307 V  2393  4771#
OLDBDOS         0B6F L   308   324#
OLDDSK          2314 V  4147  4594  4783#
ON              FFFF N    42#
OPEN            1B42 L  3312# 3927  4213
OPEN1           1B4A L  3317# 4238
OPENCOPY        1B4C L  3320# 3595
OPENERR         1CD2 L  3571  3582  3588  3601#
OPENMOD         1CA1 L  3550  3583#
OPENR0          1CAC L  3559  3574#
OPENR1          1CC7 L  3562  3578  3593#
OPENR2          1CCA L  3589  3596#
OPENRCNT        1B82 L  3356  3358  3360#
OPENREEL        1C74 L  3533# 3638  3829
OPENUSERZERO    206B L  4232#
OPENX           2081 L  4214  4239  4243#
OPENX0          20A0 L  4250  4254  4258#
PAGO            22D4 V   718   728   846  4710#
PARAMETERSEGMENT 249D V  267   320   406   437   779   945  2165  2172  2740  4276
                        4307  4876#
PARCOPFL        2327 V   329   399  4600  4806#
PARLG           2328 V   400   435  4807#
PARRET          0C03 L   355   412#
PARSAVE         0BDC L   391#  425  1257
PARSAVE33       0C04 L   414# 4117  4274
PARSAVE33R      1FA5 L  4116# 4205  4267  4310  4327  4338  4353  4368  4383  4447
PARSAVE36       0C09 L   420# 4121  4546
PARSAVE36R      1FAA L  4120# 4502  4518  4532  4618
PARUNSAVE       0C11 L   429# 4602
PCTLH           147D L  1858  1862#
PDOLLAR         199F L  2965  2972  2976#
PERERR          2204 V  2368  4691#
PERMSG          23AB V   352  4850#
PERSUB          0BA4 L   351# 4691
PHYS1           12D8 L  1558# 1565
PHYSERR         12EA L  1562  1568#
PHYSEX          12EC L  1567  1570#
PRINT           14A4 L   386   388  1893# 1909  2166
PSCOMMON        0C0C L   418   424#
PUNCHF          2512 L    90#  205
QSELECT         20CD L  4282  4291#
RANCLOSE        1E87 L  3899  3911#
RANREC          0021 N   153# 3857  4028  4073
RBLOK           1AD8 L  3190  3203  3207#
RBLOK0          1AE2 L  3196  3219#
RCOUNT          2316 V  2503  2522  3630  3795  4790#
RDBUFF          1670 L  2350# 2710  3659
RDDIR           18B4 L  2765# 2954  3033
RDDIR0          18C8 L  2774  2779#
RDDIRBUF        186C L  2706# 2795
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
RDECH1              157A L  1967  2068#
RDECH0              1574 L  2064#
RDREAD              1EDA L  3969# 4504
RDWRITE             1EE5 L  3980# 4519
READ                14B9 L  1911# 2000  2013  2173
READBUF             0C30 L   501#  748   793
READEN              159E L  1934  2093#
READERF             2515 L    91# 2116
READF               2527 L    97# 2353
READNO              14CC L  1907# 1930
READNX              14CA L  1923# 1944  1959  1987  2091
RECCNT              000F N   149# 2489  2491  3341  3362  3475  4042
RECORDOK            1D06 L  3632  3646#
RECSIZ              0080 N   133#  135  2529  2738
REMEX               11A4 L  1315  1325#
REMOVESTACK         10F8 L  1207  1214# 1234
REMOVESTACK1        1111 L  1217  1225#
REMOVETABLEELEME    1187 L  1291  1305# 1494  1543
RENAMO              1B17 L  3274# 3288
RENAME              1B04 L  3255# 4384
REP0                1547 L  2030# 2041
REP1                155A L  2032  2043#
RESEL               2313 V   332  4134  4579  4787#
RESELECT            1FAD L  4118  4124# 4289
RESELECT0           1FC5 L  4130  4133#
RESTORESTACK        0F4C L   918   927   936# 1755
RET0                13B7 L  1683  1692  1704  1706  1708  1710  1712# 1722
RET1                1449 L  1800  1803  1808  1816# 1820
RET10               1B1C L  3277# 3292# 3303  3318
RET11               1B8D L  3365# 3372
RET12               1B98 L  3376# 3384  3389  3394
RET13               1C34 L  3493# 3508  3542
RET2                1474 L  1853# 1879  1898
RET20               1FF6 L  4101  4158#
RET30               204C L  4219# 4224  4245
RET4                1683 L  2300  2337# 2367
RET5                17FA L  2580  2589  2605#
RET6                181E L  2641# 2646  2672
RET7                185E L  2684  2688  2694#
RET71               1900 L  2792  2833#
RET8                18AB L  2753  2756#
RET8B               19BF L  2956  3004#
RET9                1A55 L  3112# 3155
RETAO               0D40 L   630   637   643#
RETAFF              0D3D L   597   640#
RETM1               218E L  4601  4603#
RETMON              21B4 L  2140  4580  4591  4599#
RETMONRET           21C3 L  4605# 4623
RETRNLS             2325 V   305   313   975  4805#
RETSELECT           167F L  2330  2334#
RIGHTIST            1AC2 L  3182  3194#
RMF                 230C V  3580  3624  3675  4773#
RODERR              2208 V  2590  4693#
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
RODMSG          23C2 V   365  4855#
RODSK           22E8 V  2551  2559  2561  4187  4434  4568  4745#
RODSUB          0BB5 L   364# 4693
ROFER           220A V  2581  4694#
ROFILE          0009 N   156# 2576
ROFMSG          23BD V   369  4854#
ROFSUB          0BBA L   368# 4694
ROLDATE         24BB V  4907#
ROLTIME         24BD V  4908#
ROTR            190A L  2848# 3213
RSEEK           1E2F L  3847# 3974  3984
RSEEK1          1E34 L  3853# 4622
RUBOUT          007F N   117# 1799  1955
SAVEDCNTPOS2    1A79 L  3089  3134#
SAVESTACKS      0B1E L   274   279#
SBASE           0003 N   484#  872
SCAN3           1948 L  2905  2914#
SCNDM           1913 L  2867# 2980  3164
SCNDM0          191C L  2877# 2919
SCNDM1          1932 L  2887  2894#
SCNDM2          1939 L  2893  2901#
SEARCH          19D2 L  3015#
SEARCHI         19C0 L  3007# 3019  4236
SEARCHI1        19C8 L  3011#
SEARCHN         19DB L  3025# 3143  4237
SEARCHNJMP      1A8A L  3116  3118  3120  3136  3139  3142#
SEARCHUSER0     2397 V  3093  3119  4210  4215  4222  4838#
SECTP1          22FA V  2322  2387  4763# 4773
SECTRAN         2530 L   100# 2397
SEEK            16BA L  2381# 3658  3775  3789
SEEKCOPY        1AFE L  3247# 3437
SEEKDIR         16AA L  2371# 2794  3249
SEEKERR         1ED1 L  3886  3960#
SEEKOK          1EC3 L  3910  3931  3946#
SEGLD           0C7A L   549#  603
SELDSKF         251B L    93# 2296
SELECTDISK      1627 L  2288# 4094
SELER           2206 V  2263  4692#
SELERROR        1616 L  2261# 4097
SELMSG          23B6 V   361  4852#
SELSUB          0BB0 L   360# 4692
SEQIO           230F V  2513  3620  3670  3755  3823  3852  4620  4782#
SETALLOCBIT     1901 L  2837# 2910
SETBASE0        0DA8 L   704#  804
SETBASEA        0DC4 L   714   720#
SETBASEB        0DCA L   723#  740
SETBASEC        0DD7 L   727   729#
SETBASED        0DE8 L   725   734   736#
SETBASEE        0DEB L   731   738#
SETBASEEND      0DF0 L   719   741#
SETCDISK        17A0 L  2538# 2560  4106
SETCDR          181F L  2643# 2981  3524
SETDATA         1872 L  2703  2714# 3787  4413
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
SETDIR          187E L  2700  2708  2721# 3772
SETDMA          1886 L  2719  2727#
SETDMABASE      0033 N   464#  738
SETDMAFN        001A N   463#  791   808
SETDMBF         2533 L   101#  967  2717  2724
SETDMF          2524 L    96#  2730
SETENDDIR       13AC L  2758#  2777  2949  3020  3126
SETFC1          1783 L  2515  2517#
SETFCB          1777 L  2510#  3660  3828  3844
SETFW           180C L  2623#  3334  3528  3606  3965
SETIOBF         253C L   104#  2157
SETLRET1        15F8 L  2188#  3604  3663  3688
SETRANDOM       1F60 L  4067#  4548
SETRO           17BA L  2557#  2689  4424
SETSECF         2521 L    95#  2399
SETSIZE         1F5E L  4039  4063#
SETTRKF         251E L    94#  2394
SETUSRCODE      2152 L  4479  4486#
SFUNC86         002F N   236#  337
SHELLTOBDOS     0C42 L   506   511#  522   789   792   809
SINGLE          2312 V  2327  2440  2886  3405  3728  4785#
SLECT           1F75 L  4085#  4114  4192
SLEN1           0E1C L   768   771#
SLENGTH         0E08 L   758#  879
SPSAVE          2496 V   231   290   318   785   901  4870#
SQREAD          1CD8 L  3618#  4340
SQWRITE         1D1D L  3668#  4355
SRCH            19D2 L  3016#  3151  3266  3300  3315  3391  3505  3576  4025  4292
SRCHA           2321 V  3010  3044  4803#
SRCHASEG        24AB V  4277  4306  4889#
SRCHEXT         1A25 L  3074  3080#
SRCHFIN         1A6D L  3038  3124#
SRCHL           24A8 V  3013  3058  4837#
SRCHLOOP        1A08 L  3060#  3102
SRCHN           19DB L  3026#  3075  3166  3286  3309  4061  4311
SRCHN1          19ED L  3035  3040#
SRCHNEXT        19FF L  3048  3056#
SRCHNN          1A56 L  3078  3114#
SRCHOK          1A3C L  3066  3070  3079  3096#  3122
SSIZE           0050 N    56#  4860
SSSAVE          2494 V   230   291   317   783   899  4869#
STACKAREA       23EE V  4860#
STACKERR        10F5 L  1210#  1218
STACKFLAG       2498 V   282   513   897   937  4871#
STARET          15F4 L   242  2107  2117  2135  2144  2151  2181#  2191  2988  3131
                        3599  3721  3949  4403
STBASE          0000 N   991  1089  1171  1197  1219  1220  1226  1227  4827#
STHLRET         2141 L   247   977  4394  4421  4435  4465#
STLEN           0002 N   993  1087  1198  1202  1204  1216  1221  1223  1229  1230
                        4828#
STRLEN          0CA1 L   563   565#
STRTCOL         22E4 V  1878  1914  1978  1996  4734#
SUBDH           1828 L  2653#  2671  2773
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
SUBFLAG          24B2 V  4898#
SUBMIT           0805 V    71# 1744
TAB              0009 N   118# 1707  1843
TABO             1468 L  1847# 1852
TABENDSEG        0000 N  1421# 1429  1449  1486  1505  1525  1602  1631
TABOUT           1463 L  1696  1829  1841# 1906  2109
TAV1             12FF L  1587# 1600  1603
TAV2             131C L  1595  1601#
TAVEX            1327 L  1606  1609#
TAVFOUND         1321 L  1598  1604#
TAVNOT           1325 L  1590  1607#
TESTATSEGEND     1330 L  1119  1615#
TESTATSEGSTART   133B L  1125  1139  1641#
TESTF            0000 N    46#
TESTIFEX         1245 L  1448  1451  1453#
TESTWITHIN       1229 L  1030  1067  1434# 1563
TESTWITHINAVAIL  12F2 L  1111  1576#
TESTWITHINPHYSIC 12C7 L  1109  1547#
THEBIOS          160E L   939  2197  2203  2207  2212#
TOCONINF         1601 L  1685  1725  1729  2143  2199#
TOCONOUF         160B L  1786  1860  1865  2133  2209#
TOCONSTF         1606 L  1720  2134  2137  2205#
TODIRTOUSER      20E8 L  4293  4313#
TOLISTF          15FC L   205  1792  2195#
TOREADNO         150C L  1979# 2061
TORESTORE        13FC L  1745  1752#
TOSTARET         211E L  4257  4402# 4432
TRANV            2309 V  2311  2396  4776#
TRUE             FFFF N    44#   45   276   399   545   649   713   882  1748  2328
                        2335  2664  2953  3088  3117  3119  3121  3486  3624  3973
                        4134  4210  4215  4600
TRYZERO          204D L  4218  4221#
TYPSEG           0000 N   474#  550   552   636   724   726   733   735   815   865
UBYTES           000D N   143# 3069  3521
UMEMBASE         22D7 V   521   560   712  1090  1091  4718#
UMEMLG           22D9 V   536   580   606   614   618  1088  4719#
USERENDSEG       0002 N  1422# 1426  1450  1485  1506  1523  1596  1657
USERIPO          0005 N   485#  856
USERPARMSEG      2323 V   504   623   780  4804#
USERZEROPASS     239A V  3030  3088  3121  4841#
USRCODE          24A5 V  2963  4155  4207  4481  4488  4883#
VRECORD          2318 V  3641  4792# 4793
W                0000 N    52# 2268  2947  3371
WAITERR          0BBD L   362   366   371#
WBOOTF           2503 L    85#  938
WFLAG            230D V  2357  3753  3760  3779  4779#
WRBUFF           169D L  2358# 2702
WRBUFFLG         1699 L  2356# 3777  3791
WRDIR            185F L  2697# 3165  3250
WRITEF           252A L    98# 2364
XCHECKFCB        2005 L  4163  4166#
XDCNT            2398 V  3141  4226  4839# 4840
ZEROREADRD       0060 N   159# 4132  4162  4231  4584
```

COPYRIGHT © 1981, 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____